The Java CoG Kit User Manual
Draft Version 1.1
MCS Technical Memorandum
ANL/MCS-TM-259

Revisions March 14, 2003, July 18, 2003

*Gregor von Laszewski, Beulah Alunkal, Kaizar Amin,
Jarek Gawor, Mihael Hategan, Sandeep Nijsure

Argonne National Laboratory
Mathematics and Computer Science Division
9700 S. Cass Ave
Argonne, IL 60439

* Coresponding Author
(630) 252 0472
gregor@mcs.anl.gov

Location of Manual:

http://www.globus.org/cog/manual-user.pdf

Be kind to your environment and
do not print
this frequently changing manual.

(c) Argonne National Laboratory. All rights reserved.

January 30, 2004

Contents

1 License **8**

1.1 General Comments 8

1.2 Globus Toolkit Public License (GTPL) 9

1.3 Other Licences 11

1.3.1 jglobus 11

1.3.2 ogce 11

2 Preface **12**

2.1 Intended Audience 12

2.2 Resources 13

2.2.1 Project Website 13

2.2.2 Bug Reporting 13

2.2.3 Mailing Lists 13

2.2.4 Sourcecode Repository 14

2.3 Manual Guidelines 15

2.3.1 Conventions 15

2.3.2 Contributions 16

2.4 Administrative Contact 16

2.5 Acknowledgments 16

3 Installation **17**

3.1 Introduction 17

3.2 Requirements 17

3.2.1 Java Development Kit 17

3.2.2 Ant 17

3.3 Java CoG Kit Formats 17

3.3.1 The Java CoG Kit Parts 18

3.3.2 Stable and Development Distributions 18

3.3.3 Formats 18

3.3.4 What to Choose 18

3.4 Downloading the Java CoG Kit 19
3.4.1 JGlobus Stable Binary 19
3.4.2 JGlobus Stable Source 20
3.4.3 JGlobus Development Source 20
3.4.4 OGCE Stable Source 21
3.4.5 OGCE Development Source 21
3.5 Compiling 21
3.5.1 Compiling JGlobus 21
3.5.2 Compiling OGCE 22
3.6 Configuration 22
3.6.1 Environment Variables 22
3.6.2 Time Synchronization 23
3.6.3 Globus Security Credentials 23
3.6.4 Configuration 23

4 Security **25**
4.1 Introduction 25
4.1.1 Grid Security Infrastructure 25
4.1.2 Certificates and certifying authorities 26
4.1.3 Proxies and delegation 26
4.2 Security prerequisites 27
4.2.1 Acquiring a user certificate 27
4.2.2 Acquiring a host certificate (optional) 27
4.2.3 Renewing a certificate 28
4.2.4 Obtaining the certificates of the trusted CAs 28
4.2.5 Gridmap files 28
4.2.6 Protecting credentials 28
4.3 MyProxy 29
4.4 Managing certificates and proxies 30
4.4.1 GUI 30
4.4.2 Unix shell scripts 30
4.4.3 Windows batch files 32
4.4.4 Java CoG Kit shell 33
4.4.5 API 33
4.5 Firewall Issues 36

4.6 Random number generation issues . . . 37

5 File I/O and Transfer **38**

5.1 Introduction . . . 38

5.1.1 Requirements for File Access and Transfer over the Grid . . . 38

5.1.2 GridFTP . . . 38

5.1.3 GASS . . . 39

5.1.4 Other file transfer mechanisms . . . 40

5.1.5 Security Requirements . . . 40

5.2 Using GridFTP . . . 40

5.2.1 GUI . . . 40

5.2.2 Unix Shell Scripts . . . 42

5.2.3 Windows Batch Files . . . 43

5.2.4 Java CoG Kit Shell . . . 43

5.2.5 APIs . . . 43

5.2.6 Differences between Java CoG Kit version 0.9.13 and 1.1a 46

5.2.7 FTP/GridFTP protocol features supported by the Java CoG Kit . . . 46

5.2.8 Limitations of the Java CoG Kit . . . 47

5.3 Using GASS . . . 47

5.3.1 GUI . . . 47

5.3.2 Unix Shell Scripts . . . 47

5.3.3 Windows Batch Files . . . 48

5.3.4 Java CoG Kit Shell . . . 48

5.3.5 APIs . . . 48

5.3.6 Limitations of the Java CoG Kit . . . 51

6 Job Submission **52**

6.1 Introduction . . . 52

6.1.1 Gatekeeper . . . 52

6.1.2 Job Manager . . . 52

6.1.3 Batch and Interactive Jobs . . . 53

6.1.4 File Staging . . . 53

6.2 Globus Resource Specification Language (RSL) . . . 53

6.2.1 RSL Syntax . . . 54

6.2.2 RSL in the Java CoG Kit . . . 55

6.3 Job Submission 55
6.3.1 GUI 55
6.3.2 Unix Shell Scripts 56
6.3.3 Windows Batch Files 57
6.3.4 Java CoG Kit Shell 57
6.3.5 Job Submission API 57
6.4 Differences from the C Globus Toolkit 59
6.4.1 Gatekeeper 59
6.4.2 RSL Parser 59

7 Accessing the Grid Information Service 60
7.1 Introduction 60
7.2 Architecture 60
7.2.1 GRIS 60
7.2.2 IPs 61
7.2.3 GIIS 61
7.2.4 Working 61
7.3 Security with MDS 61
7.3.1 Site Policies 61
7.4 Accessing Grid Information Services 62
7.4.1 Using Graphical User Interface (GUI) 62
7.4.2 Unix Shell scripts 62
7.4.3 Windows batch files 64
7.4.4 Using the API to access MDS 64
7.5 Schema 68
7.6 Performance issues with MDS 69
7.6.1 Programming Issues 70
7.7 Implementation Details of MDS 2.2 version 71
7.8 Differences between Java and Globus tool 71

8 Server-side Java CoG Kit 72
8.1 Introduction 72
8.2 Job Execution Service 72
8.2.1 Configuration 73
8.2.2 Limitations 73

8.2.3 Starting the personal gatekeeper . . . 73
8.2.4 Differences between Java and Globus Personal Gatekeeper 74
8.3 File Transfer Service . . . 75
8.3.1 Limitations . . . 75
8.3.2 Starting the Gass Server . . . 75
8.3.3 Differences between Java and Globus GASS service . . . 76

9 Production Tests with the Java CoG Kit 77
9.1 Introduction . . . 77
9.2 Requirements . . . 77
9.3 Installation . . . 78
9.4 Configuration . . . 78
9.5 Host Table Format . . . 79
9.6 Running the Tests . . . 80

10 GridAnt: A Client-side Grid Workflow System 82
10.1 Introduction . . . 82
10.2 GridAnt Tasks . . . 83
10.2.1 gridExecute . . . 83
10.2.2 gridCopy . . . 84
10.3 Installation . . . 85
10.4 Security . . . 86
10.5 Examples . . . 86
10.5.1 gridExecute . . . 86
10.5.2 gridCopy . . . 86
10.6 Complex Example . . . 86

A Program Options 87
A.1 globus2jks . . . 87
A.2 globus-gass-server . . . 87
A.3 globus-gass-server-shutdown . . . 88
A.4 globus-personal-gatekeeper . . . 88
A.5 globusrun . . . 89
A.6 globus-url-copy . . . 90
A.7 grid-cert-info . . . 91
A.8 grid-change-pass-phrase . . . 91

A.9 grid-info-search 92
A.10 grid-proxy-destroy 93
A.11 grid-proxy-info 93
A.12 grid-proxy-init 94
A.13 myproxy 95

B Command overview **96**
B.1 New Format for the table 96

C Frequently Asked Questions **97**
C.1 Installation 97
C.2 Security 97
C.2.1 General Grid security Questions 97
C.2.2 Questions related to user certificates and certificate authority 98
C.2.3 Questions related to proxy certificates 98
C.2.4 Questions related to host certificates and gridmap files . . 98
C.2.5 MyProxy 98
C.2.6 Miscellaneous 98
C.3 File I/O and Transfer 99
C.3.1 Overview 99
C.3.2 GridFTP 99
C.3.3 GASS 100
C.3.4 Version differences 100
C.4 Job Execution 100
C.4.1 GRAM 100
C.5 Grid Information Service 101
C.5.1 General 101
C.5.2 Architecture of MDS 101
C.5.3 Security in MDS 102
C.5.4 Retrieving information from MDS 102
C.5.5 Performace Issues with MDS 102
C.6 Server Side Java CoG Kit 102
C.6.1 General 102
C.6.2 Job Execution Service 103
C.6.3 GASS server 103

C.7 GridAnt 104

1 License

1.1 General Comments

The Java CoG Kit is distributed under the Globus Toolkit Public License (GTPL), which is listed in Section 1.2.

We kindly ask you to notify us about projects that you develop with the help of the Java CoG Kit. This will allow us to keep track of the use of the Java CoG Kit, as this directly affects our ability to motivate additional coding activities. Please, be so kind to send an e-mail to gregor@mcs.anl.gov with the subject JAVA COG KIT USGAE or fill out a form at

Form : http://www-unix.globus.org/cog/projects/add/

with the following description:

Project name:
Institution:
Main contact:
E-mail:
Web page:
Description of your project:
References:
References citing the Java CoG Kit:

In case you like to cite the Java CoG Kit in your papers, we recommend that you use the following paper:

Gregor von Laszewski, Ian Foster, Jarek Gawor, Peter Lane,
A Java Commodity Grid Kit,
Concurrency and Computation: Practice and Experience,
Pages 643-662, Volume 13, Issue 8-9, 2001.
http://www.globus.org/cog/java/

We also would like to be notified about your publications that involve the use of the Java CoG Kit, as this will help us to document its usefulness. We like to feature links to these articles, with your permission, on our Web site.

Additional references to Java CoG Kit and other Grid related activities can be found at

Some Refernces, von Laszewski : http://www.mcs.anl.gov/~gregor/bib

or

Some References, Globus Project : http://www.globus.org/research/papers.html

.

1.2 Globus Toolkit Public License (GTPL)

Copyright (c) 1999 University of Chicago and The University of Southern California. All Rights Reserved.

1. The “Software”, below, refers to the Globus Toolkit (in either source-code, or binary form and accompanying documentation) and a “work based on the Software” means a work based on either the Software, on part of the Software, or on any derivative work of the Software under copyright law: that is, a work containing all or a portion of the Software either verbatim or with modifications. Each licensee is addressed as “you” or “Licensee.”

2. The University of Southern California and the University of Chicago as Operator of Argonne National Laboratory are copyright holders in the Software. The copyright holders and their third party licensors hereby grant Licensee a royalty-free nonexclusive license, subject to the limitations stated herein and U.S. Government license rights.

3. A copy or copies of the Software may be given to others, if you meet the following conditions:

   (a) Copies in source code must include the copyright notice and this license.

   (b) Copies in binary form must include the copyright notice and this license in the documentation and/or other materials provided with the copy.

4. All advertising materials, journal articles and documentation mentioning features derived from or use of the Software must display the following acknowledgement:

   ”This product includes software developed by and/or derived from the Globus project (http://www.globus.org/).”

   In the event that the product being advertised includes an intact Globus distribution (with copyright and license included) then this clause is waived.

5. You are encouraged to package modifications to the Software separately, as patches to the Software.

6. You may make modifications to the Software, however, if you modify a copy or copies of the Software or any portion of it, thus forming a work based on the Software, and give a copy or copies of such work to others, either in source code or binary form, you must meet the following conditions:

   (a) The Software must carry prominent notices stating that you changed specified portions of the Software.

   (b) The Software must display the following acknowledgement:

   “This product includes software developed by and/or derived from the Globus Project (http://www.globus.org/) to which the U.S. Government retains certain rights.”

7. You may incorporate the Software or a modified version of the Software into a commercial product, if you meet the following conditions:

(a) The commercial product or accompanying documentation must display the following acknowledgment:

"This product includes software developed by and/or derived from the Globus Project (http://www.globus.org/) to which the U.S. Government retains a paid-up, nonexclusive, irrevocable worldwide license to reproduce, prepare derivative works, and perform publicly and display publicly."

(b) The user of the commercial product must be given the following notice:

"[Commercial product] was prepared, in part, as an account of work sponsored by an agency of the United States Government. Neither the United States, nor the University of Chicago, nor University of Southern California, nor any contributors to the Globus Project or Globus Toolkit nor any of their employees, makes any warranty express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

IN NO EVENT WILL THE UNITED STATES, THE UNIVERSITY OF CHICAGO OR THE UNIVERSITY OF SOUTHERN CALIFORNIA OR ANY CONTRIBUTORS TO THE GLOBUS PROJECT OR GLOBUS TOOLKIT BE LIABLE FOR ANY DAMAGES, INCLUDING DIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM EXERCISE OF THIS LICENSE AGREEMENT OR THE USE OF THE [COMMERCIAL PRODUCT]."

8. LICENSEE AGREES THAT THE EXPORT OF GOODS AND/OR TECHNICAL DATA FROM THE UNITED STATES MAY REQUIRE SOME FORM OF EXPORT CONTROL LICENSE FROM THE U.S. GOVERNMENT AND THAT FAILURE TO OBTAIN SUCH EXPORT CONTROL LICENSE MAY RESULT IN CRIMINAL LIABILITY UNDER U.S. LAWS.

9. Portions of the Software resulted from work developed under a U.S. Government contract and are subject to the following license: the Government is granted for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable worldwide license in this computer software to reproduce, prepare derivative works, and perform publicly and display publicly.

10. The Software was prepared, in part, as an account of work sponsored by an agency of the United States Government. Neither the United States, nor the University of Chicago, nor The University of Southern California, nor any contributors to the Globus Project or Globus Toolkit, nor any of their employees, makes any warranty express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

11. IN NO EVENT WILL THE UNITED STATES, THE UNIVERSITY OF CHICAGO OR THE UNIVERSITY OF SOUTHERN CALIFORNIA OR ANY CONTRIBUTORS TO THE GLOBUS PROJECT OR GLOBUS TOOLKIT BE LIABLE FOR ANY DAMAGES, INCLUDING DIRECT, INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM

EXERCISE OF THIS LICENSE AGREEMENT OR THE USE OF THE SOFTWARE.

END OF LICENSE

1.3 Other Licences

We distribute a number of other libraries with the Java CoG Kit. These libraries come with their own licences. We strongly encourage you to inspect these licenses. The can be found in the "lib" directories of the Java CoG Kit.

1.3.1 jglobus

The jglobus/lib directory contains the following licences.

jglobus : bouncycastle.LICENSE

jglobus : cryptix.LICENSE

jglobus : log4j.LICENSE

jglobus : junit.LICENSE

jglobus : puretls.LICENSE

1.3.2 ogce

The ogce/lib directory contains the following licences:

ogce : soaprmi11.LICENSE

ogce : xerces.LICENSE

ogce : xml4j.LICENSE

2 Preface

Grids are an important development in the discipline of computer science and engineering. Rapid progress is being made on several levels, including the definition of the terminology, the design of an architecture and framework, the application in the scientific problem solving process, and the creation of physical instantiations of Grids on a production level.

A small overview about the Grid can be found in a draft paper entitled *Gestalt of the Grid*

Article : http://www.mcs.anl.gov/~gregor/bib/papers/vonLaszewski--gestalt.pdf

This article provides an overview of important influences, developments, and technologies that are shaping state-of-the-art Grid computing. In particular, we address the following questions:

What motivates the Grid approach? What is a Grid? What is the architecture of a Grid? Which Grid research activities are performed? How do researchers use a Grid? What will the future bring?

Other CoG Kit related papers can be found at

References von Laszewski : http://www.mcs.anl.gov/~gregor/bib/

2.1 Intended Audience

This manual is intended for the intermediate Grid programmer that would like to access the Globus Toolkit functionality through Java. We assume that the reader of this manual is familiar with Java. If not, general information about Java is available through the Web site at SUN Microsystems or at IBM:

SUN : http://java.sun.com/

IBM : http://www.ibm.com/java/

In general, this manual serves as a basic introduction to a subset of functionality provided by the Java CoG Kit. This manual does not explain every package, class, and method. This manual is intended to show you that the Java CoG Kit provides an effective way of accessing the Grid through Java.

Developers are encouraged to inspect the JavaDoc documentation.

We further expect that you are familiar with the Globus Toolkit and have access to a Globus Toolkit 2 installation. If you do not, the Globus web page provides information about the details and how to install it.

Globus Toolkit : http://www.globus.org

2.2 Resources

We support our efforts through a web site on which you find a bug tracking system, Mailing lists, and the code repository.

2.2.1 Project Website

Online information about the Java CoG Kit can be found on its home page.

Home page : `http://www.globus.org/cog/java/`

Here you can find links to the manual, the code, and some basic information about the project. Besides this page we also maintain a project-related Web page that reports on the Java and Python Commodity Grid Kits.

Project : `http://www.cogkits.org/`

2.2.2 Bug Reporting

We are using the Bugzilla system from mozilla.org to track bugs and requests for enhancements for the Java CoG Kit. Bugzilla provides you with an interface that guides you on submitting the bug. The link to the bug system is located at

CoG Kit Bugzilla : `http://www.globus.org/cog/contact/bugs/`

In case you like to report bugs for other components of the Globus Toolkit you can use the main link at

Globus Toolkit Bugzilla : `http://bugzilla.globus.org/globus/`

To use it you need to first create an account. To report a bug you need to be precise in your description and include operating system, JVM version, and other information that can be used to better identify or replicate the condition of your error. This also includes the version of Globus Toolkit services you use.

2.2.3 Mailing Lists

We have established a number of mailing lists to simplify the communication with the group of developers and users. Restrictions on the use of the mailing list are outlined below.

Policy

No Advertisements : We do not allow you to use the mailing lists in any form of advertisement for your products or services. In response to spam mail on this mailing list, we have disabled the ability to post messages to this list if you are not subscribed to it.

Subscription Required : If you send a message to the list and are not subscribed or you use an email address different from the one you subscribed with, your message will not be posted to the list, and you will not receive any notification that your message was *not* posted. Hence, if you send a message to the list and do not subsequently see your message on the list or in the list archive, verify that you are using an email address that is subscribed to the list, and then retry your posting.

Subscribed Lists : To verify that you are subscribed to the list, send an email message from the email account you subscribed from to majordomo@globus.org with the single word "which" in the body of the message. You will receive in response a message listing the lists to which your email address is subscribed. If this mailing list does not appear in the list you receive, you are probably subscribed to the list under a different address and you will not be able to post messages to the list using your current address.

Subscription Center

If you would like to be notified of CoG Kit release updates, visit our convenient subscription center at

Subscribe : http://www.globus.org/cog/contact/

Other Globus related mailing lists can be found on the Globus web page

Subscribe : http://www.globus.org/about/subscriptions.html

Note that you can use these web pages to unsubscribe from the lists. All mailing list are maintained with majordomo. However, we did have to disable the *who* function in order to protect the members from spam bots.

News

News about the Java CoG Kit is sent in irregular intervals (the frequency is monthly to every four month) by means of the following list:

CoG News : cog-news@globus.org

Sorted by Thread: : http://www-unix.globus.org/mail_archive/cog-news/threads.html

Sorted by Date : http://www-unix.globus.org/mail_archive/cog-news/maillist.html

Discussions and Community Developers

Discussions and general questions can be send to the high-volume e-mail list at

Java List : java@globus.org

Sorted by Thread : http://www-unix.globus.org/mail_archive/java/threads.html

Sorted by Date: : http://www-unix.globus.org/mail_archive/java/maillist.html

Note that this list may result in daily mails sent by the Java CoG Kit community. Please use the bug tracking system for reporting bugs. If you use the bug tracking system, your message has a higher chance of being answered. There is no guarantee that we answer a mail sent to the Java CoG Kit mailing lists.

2.2.4 Sourcecode Repository

We maintain all source code in a CVS repository that can be accessed anonymously. You can find more details about this in Section 3.4.5.

2.3 Manual Guidelines

This manual is constantly being improved and your input is highly appreciated. Please report suggestion, errors, changes, and new sections or chapters to this document to

Gregor von Laszewski : gregor@mcs.anl.gov

When you report bugs, please do not use page, line, or section numbers. Remember new sections may appear due to community contributions. Instead, please quote the section title, or make corrections by hand and FAX it to us. Even better, submit a corrected document, as you can check out the manual through our CVS archive.

2.3.1 Conventions

If you see a ?? or a ... in the text there is no reason to send us a report on it. It simply means that the section to which we refer has not yet been integrated in this manual. Regular text is written using the Times font. Code examples use the

```
Courier
```

font. For code example contributions, we recommend not exceeding the margin width of the paper and make the lines no longer than 79 characters. An example is shown below.

```
int a;
a = 1+ 2;
```

Interactive commands issued by a user in a shell are preceded with a $>$ at the beginning of the line.

```
> mkdir directory
> cd directory
```

In case interactive commands exceed the 79 character limit, they are wrapped into the next line and are not proceeded by the $>$ character. A backslash is included at the end of such lines to explicitly indicate that the command ins continued on the next line.

```
> echo "This is s very long text that is continued on the next \
        lines. The leading blanks in the next lines are to \
        be ignored"
> echo "This is a new command"
```

References to variables or other important text that is part of a program or shell script is written in `Courier`. To illustrate this on an example:

> Hence, a reference to the variable `int a` form our previous example uses also the `Courier` font.

Generic entities are wrapped between angle brackets. Each such entity is not to be taken literally. In general, such constructs are explained as they occur throughout the manual. The use of such entities is shown in the example below:

```
> ping <machine-name>
```

Here, <machine-name> is to be replaced with an actual machine name:

```
> ping hot.mcs.anl.gov
```

Web links are proceeded by a meaningful name for the link. An example is

Java CoG Kit Website : `http://www.globus.org/cog`

Links to code source are proceeded by the repository tag. An example is

jglobus : `org/globus/gram/Gram.java`

2.3.2 Contributions

This manual contains, in alphabetical order, contributions from Beulah Alunkal (ANL), Kaizar Amin (ANL), Jarek Gawor (ANL), Mihael Hategan (ANL), Sandeep Nijsure (ANL), Gregor von Laszewski (ANL).

Additional contributions during the course of the Java CoG Kit development have been made by (sorted in alphabetical order):

Peter Lane(ANL), Jason Novotny (LBL now MPI), Nell Rehn (ANL now IBM), Mike Russell (UC now MPI), Pawel Plaszczak (ANL), Carlos Peña (ANL now NYU), Warren Smith (ANL now NASA), Andreas Schreiber (DLRZ), Patrick Wagstrom (ANL now IIT).

If we have forgotten to include your name in the list of contributors please notify us.

We invite you to contribute to the manual or the code.

2.4 Administrative Contact

The project is managed by Gregor von Laszewski. To contact him, please use the information below.

Gregor von Laszewski
Argonne National Laboratory
Mathematics and Computer Science Division
9700 South Cass Avenue
Argonne, IL 60439
Phone:(630) 252 0472
Fax: (630) 252 1997
gregor@mcs.anl.gov

2.5 Acknowledgments

This work was supported by the Mathematical, Information, and Computational Science Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-Eng-38. DARPA, DOE, and NSF support Globus Project research and development. This work would not have been possible without the help of Ian Foster and the Globus Project team.

3 Installation

In this chapter you will learn how to to download, install, and configure the Java CoG Kit.

3.1 Introduction

Installation is the first step that needs to be accomplished before the Java CoG Kit can be used. It ensures that the Java CoG Kit exists on your local machine in a proper state. After installation, configuration is needed to adjust various parameters that are specific to your environment.

3.2 Requirements

The Java CoG Kit has a minimal installation requirement. In most cases it is only necessary to have a Java Virtual machine. In case you also like to make use of the GridAnt system you will also need ant.

3.2.1 Java Development Kit

In order to be able to compile and run the Java CoG Kit, you will need to have a recent version of the Java Development Kit. The recommended version is 1.4.1. The minimum required version of the Java Development Kit is 1.3.1[1]

JDK : `http://java.sun.com`

3.2.2 Ant

The Java CoG Kit uses the Apache Ant build system. At least version 1.5.2 of Apache Ant is required by the Java CoG Kit. Please make sure that along with Ant you also install any libraries required by Ant. The Ant binaries, sources, and information about Ant requirements can be found on the Ant web-site [1]

Ant : `http://ant.apache.org`

3.3 Java CoG Kit Formats

The Java CoG Kit contains two major parts: jglobus and ogce. The Java CoG Kit is available in a number of formats that address different categories of users. In the following sections we will try to explain which part and version is suitable for a certain type of user.

[1] Please note that if you do not plan to compile the Java CoG Kit yourself, you could just use the Java Runtime Environment. The version requirements still apply.

3.3.1 The Java CoG Kit Parts

jglobus

JGlobus contains just the basic components and API's to interface with GT2.0 and GT3.0.

OGCE

OGCE[2] contains possible future enhancements and showcases that use of some of the features of jglobus.

3.3.2 Stable and Development Distributions

Stable Distribution

The stable distribution is recommended for production environments. It comes in two formats: binary and source.

Development Distribution

The development distribution contains the latest features of the Java CoG Kit, but without being tested extensively. The development version is only available from the source repository.

3.3.3 Formats

Java CoG Kit Binaries

The binary format of the Java CoG Kit requires minimal effort for the installation process. It is prepackaged in both `tar.gz` and `zip` archives.

Java CoG Kit Sources

The Java CoG Kit sources are available for users who wish to compile the Java CoG Kit themselves, or wish to see the sources of the Java CoG Kit.

Java CoG Kit Source Repository

The source repository contains the absolute latest version of the Java CoG Kit.

3.3.4 What to Choose

We identified a list of possible types of users, which may help you quickly decide which version is best for you:

Normal Users : Users who want to use stable and tested Java CoG Kit tools and do not plan to modify or extend the Java CoG Kit.

Developers : Users who want to integrate the features of the Java CoG Kit inside their own Grid applications, while using the Java CoG Kit APIs.

Contributors : Users who want to extend the features of the Java CoG Kit.

GT3 Users : Users that will use the GT3 distribution

[2] OGCE stands for Open Grid Computing Environment

A pictural representation of a mapping between the various user types and the Java CoG Kit distributions is provided in Figure 3.1.

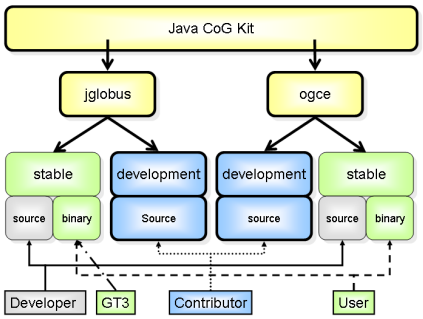

Figure 3.1: Distribution chart of the Java CoG Kit.

The following summary may help you further in your decision on which version you need to obtain and how to proceed. Following each item, a link to the section that describes details for that item is provided.

For users only interested in jglobus, the following choices are available:

jglobus stable binary : Users that are interested in just the jar files without modifying them (Section 3.4.1).

jglobus stable source : Users that are interested in also seeing the source to the stable binary version (Section 3.4.2).

jglobus development source : Users that like to work with the newest version of the code (Section 3.4.3).

For users interested in OGCE, the following choices are available:

ogce stable source : Users that are interested in also seeing the source to the stable binary version (Section 3.4.4).

ogce development source : Users that like to work with the newest version of the code (Section 3.4.5).

Most users may just be interested in the stable ogce and jglobus sources distribution. Hence, we refer to the Java CoG Kit in this manual as the combined contributions presented in the jglobus and ogce directories.

3.4 Downloading the Java CoG Kit

This section instructs you on how to download various Java CoG Kit versions.

3.4.1 JGlobus Stable Binary

The stable binary distribution of the jglobus is available from our web-site:

- *tar.gz archive:*

cog-1.1-bin.tar.gz : `www.globus.org/cog/java/1.1/cog-1.1-bin.tar.gz`

- *zip archive:*

cog-1.1-bin.zip : `www.globus.org/cog/java/1.1/cog-1.1-bin.zip`

After downloading, unpack the archive:

Unix : `> tar -xzf cog-1.1-bin.tar.gz`

Windows : Double click on the downloaded archive and extract it to a directory of your choice

A directory named `cog-1.1` will be created. This directory will, from now on, be referred to as <`cog-install-path`>

You can now proceed to configure jglobus, as described in Section 3.6

3.4.2 JGlobus Stable Source

The stable source distribution of the jglobus is available from our web-site:

- *tar.gz archive:*

cog-1.1-src.tar.gz : `www.globus.org/cog/java/1.1/cog-1.1-src.tar.gz`

- *zip archive:*

cog-1.1-src.zip : `www.globus.org/cog/java/1.1/cog-1.1-src.zip`

After downloading, unpack the archive:

Unix : `> tar -xzf cog-1.1-src.tar.gz`

Windows : Double click on the downloaded archive and extract it to a directory of your choice

A directory named `cog-1.1` will be created. This directory will, from now on, be referred to as <`cog-jglobus-src`>

You can now proceed to compile jglobus, as described in Section 3.5.1.

3.4.3 JGlobus Development Source

The development version of jglobus can be retrieved from our source repository using anonymous CVS access [3].

We suggest that you first create a new directory in which to store the development version of jglobus. For convenience this directory will be referred to as <`jglobus-devel`>.

```
> mkdir <jglobus-devel>
> cd <jglobus-devel>
```

Login to the CVS server:

```
> cvs -d :pserver:anonymous@cvs.globus.org:/home/dsl/cog/CVS login
```

Hit ENTER when you are asked for a password. After the login step, you can check out the jglobus module with the following command:

```
> cvs -d :pserver:anonymous@cvs.globus.org:/home/dsl/cog/CVS \
      co -r jglobus-jgss jglobus
```

[3] You need to have CVS installed on your system before downloading the jglobus development version

Inside the <jglobus-devel> directory, another directory named jglobus will be created. This directory will be represented by <cog-jglobus-src>.

You can now proceed to compile jglobus, as described in Section 3.5.1.

3.4.4 OGCE Stable Source

The OGCE stable source is not available at this time. Please use the development OGCE source (3.4.5).

3.4.5 OGCE Development Source

The development version of OGCE can be retrieved from our source repository using anonymous CVS access[4]. Please not that jglobus is needed in order to use OGCE. This section will provide instructions to download both jglobus and OGCE. We suggest that you first create a new directory in which to store the development version of jglobus. For convenience this directory will be referred to as <cog-devel>. We recommend that you name this directory "cog".

```
> mkdir <cog-devel>
> cd <cog-devel>
```

Login to the CVS server:

```
> cvs -d :pserver:anonymous@cvs.globus.org:/home/dsl/cog/CVS login
```

Hit ENTER when you are asked for a password. After the login step, you can check out the jglobus module with the following command:

```
> cvs -d :pserver:anonymous@cvs.globus.org:/home/dsl/cog/CVS \
      co -r jglobus-jgss jglobus
> cvs -d :pserver:anonymous@cvs.globus.org:/home/dsl/cog/CVS \
      co -r jglobus-jgss ogce
```

Inside the <cog-devel> directory, two directories named jglobus and ogce will be created. These directories will be represented by <cog-jglobus-src>, respectively <cog-ogce-src>.

You can now proceed to compile OGCE, as described in Section 3.5.2

3.5 Compiling

This section will explain the steps required to compile the Java CoG Kit.

3.5.1 Compiling JGlobus

To compile jglobus, simply do the following:

```
> cd <cog-jglobus-src>
> ant dist
```

This will compile and build jglobus. The build process will create a build directory in the <cog-jglobus-src> directory. The build directory will contain all the compiled classes, the Java CoG Kit directory and a set of examples:

[4] You need to have CVS installed on your system before downloading the jglobus development version

```
<cog-jglobus-src>/build/classes
<cog-jglobus-src>/build/cog-1.1
<cog-jglobus-src>/build/cog-1.1/bin
<cog-jglobus-src>/build/examples
```

From this point on, <cog-install-path> will represent the <cog-jglobus-src>/build/cog-1.1 directory. You can now proceed to configure jglobus as shown in Section 3.6

3.5.2 Compiling OGCE

To compile ogce, simply do the following:

```
> cd <cog-ogce-src>
> ant dist
```

This will compile and build jglobus. The build process will create a build directory in the <cog-globus-src> directory. The build directory will contain all the compiled classes, the Java CoG Kit directory and a set of examples:

```
<cog-devel>/build/classes
<cog-devel>/build/cog-1.1
<cog-devel>/build/cog-1.1/bin
<cog-devel>/build/examples
```

From this point on, <cog-install-path> will represent the <cog-devel>/build/cog-1.1 directory. You can now proceed to configure jglobus as shown in Section 3.6

3.6 Configuration

This section will show you how to configure the Java CoG Kit.

3.6.1 Environment Variables

After installing, and eventually compiling, the Java CoG Kit, you will need to set the COG_INSTALL_PATH environment variable, which is used by various tools inside the Java CoG Kit to determine the installation location of the Java CoG Kit. COG_INSTALL_PATH should point to the <cog-install-path> directory. The exact value of <cog-install-path> depends on the Java CoG Kit distribution that you chose to download, and it has been explained in its respective installation subsection.

It is also highly recommended that you add the <cog-install-path>/bin directory to your binary search path (named PATH on most systems). Most of the examples in this manual assume that you have done so. If the binary search path is not updated to include the Java CoG Kit bin directory, you will have to specify the path to the Java CoG Kit bin directory when running any of the executables shown in the examples:

Unix : `> <cog-install-path>/bin/<executable>`

Windows : `> <drive-letter>:\<cog-install-path>\bin\<executable>`

3.6.2 Time Synchronization

The Java CoG Kit requires that your date and time are properly set. The recommended way to do this is by synchronizing your system clock through the NTP protocol. Please consult your system administrator about the NTP protocol and time synchronization. Alternatively, you could synchronize your system clock using one of the following methods:

Windows

NT Atomic Clock Synchronizer : `www.worldtimeserver.com`

Atomic Clock Synchronizer download : `http://www.worldtimeserver.com/atomic-clock/`

UNIX/Linux

On UNIX you can configure automatic synchronization through with a nearby NTP server.

3.6.3 Globus Security Credentials

Using the Java CoG Kit requires you to have a proper set of Globus credentials including, but not limited to, a Globus certificate. For details about Globus security credentials, please consult Section 4.2.

3.6.4 Configuration

This subsection will explain the different methods that can be used to configure the Java CoG Kit.

Configuration with the Wizard

To start the configuration wizard for the Java CoG Kit run the setup script available in the `<cog-install-path>/bin` directory. A sample screen-shot of the setup wizard is shown in Figure 3.2.

Configuration with an Editor

Manual configuration of the Java CoG Kit is also possible. The configuration file is named `cog.properties` and is located in the `<user-home>/.globus` directory. A sample Java CoG Kit configuration file is provided in Figure 3.3. It includes a number of important properties. These properties are:

usercert : points to the location of the Globus user certificate.

userkey : points to the location of the private key associated with the Globus user certificate.

proxy : points to the location of the user proxy. The proxy is located in a temporary directory, and has its name composed of the string `x509up_u` and the a user id (OS specific). In the above example, the user id is 1000.

cacert : contains a comma separated list of certificate authorities that the user trusts.

ip : represents the IP address of the machine the Java CoG Kit will be run from.

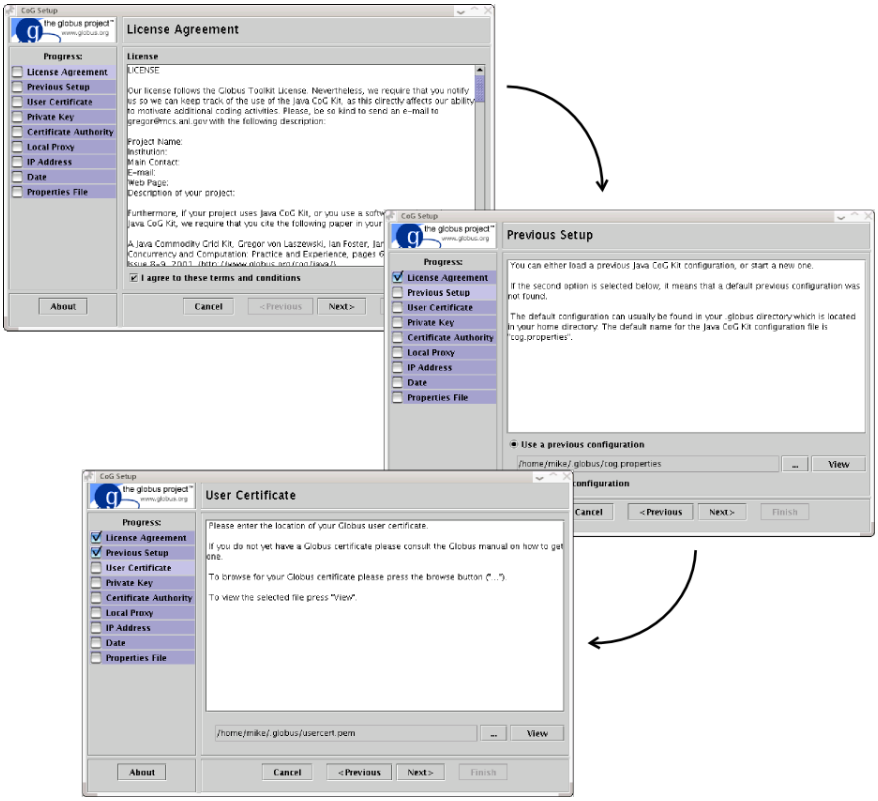

Figure 3.2: Screen-shot of the setup wizard

An additional list of properties that can but set in the `cog.properties` file, but which are not configured by the Setup Wizard is provided below:

tcp.port.range : A range of ports, in the form `<minport>, <maxport>` that limits the local ports used for services by the Java CoG Kit.

org.globus.dev.random : A `true` or `false` value specifying whether the Java CoG Kit should use the Unix style `/dev/urandom` device for random number generation.

random.provider : Specifies the Java random provider to be used by default.

random.algorithm : Specifies the random algorithm to be used for generating secure random numbers.

proxy.strength : Indicates (in bits), the default strength of the security proxy.

proxy.lifetime : Specifies the lifetime, in hours, of the security proxy.

```
#Java CoG Kit Configuration File
#Tue Feb 25 22:30:30 CST 2003
usercert=/home/albert/.globus/usercert.pem
userkey=/home/albert/.globus/userkey.pem
proxy=/tmp/x509up_u1000
cacert=/usr/local/globus/share/certificates/42864e48.0
ip=140.221.56.12
```

Figure 3.3: A sample cog.properties file for the user *albert*

4 Security

Security is of paramount importance in the Grid computing paradigm. The Globus Toolkit uses the Grid Security Infrastructure (GSI) [2] for secure access to Grid resources. Users of the Java CoG Kit thus need to interact with the GSI in order to access the Grid resources.

This chapter starts with a brief discussion about the security issues involved in the Grid paradigm and how GSI addresses these issues. It then provides an introduction to various GSI concepts like certificates, certifying authorities, proxies and gridmap authorization. The subsequent section explains the procedures to acquire the necessary credentials. A web-based credential management software called MyProxy [3] is then discussed. We then describe how a user can use the various tools that the Java CoG Kit provides for managing (creating, destroying, examining, etc.) certificates and proxies. A discussion of the security API provided by the Java CoG Kit follows. The next section describes the issues that need to be faced when using GSI across network boundaries guarded by firewalls. We conclude with a discussion of feature differences between the Java CoG Kit and the C implementations of the Globus Toolkit.

4.1 Introduction

This section assumes some knowledge of the fundamentals of information security and Public Key cryptography. If you are not familiar with these concepts, please refer to a book, such as [4]. Working knowledge of Secure Sockets Layer (SSL) [**?**] is also assumed.

4.1.1 Grid Security Infrastructure

The Grid infrastructure allows users to access computational and data resources that may span organizational, and perhaps national, boundaries. Thus it is very important to ensure that this access is secure. The basic security requirements are user authentication, confidentiality and communication integrity. Additionally, single sign-on is desired in order to allow the user to only authenticate once, irrespective of the number of resources he/she needs to access. This lets the user access resources with the least amount of manual intervention. In addition to satisfying these requirements, the security infrastructure needs to interoperate with the various security paradigms being used today in different organizations. This is necessary since it is not convenient for those organizations to abandon the existing infrastructure and switch to a new one. The Grid Security Infrastructure (GSI) [2, 5] satisfies all the requirements mentioned above.

GSI uses the Public Key Infrastructure (PKI), X.509 certificates, and Secure Sockets Layer (SSL) as its basis. It extends these standards for single sign-on. Thus, users do not have usernames/passwords in GSI. Instead they have public/private key pairs and identity certificates.

4.1.2 Certificates and certifying authorities

Every user and service on the Grid has a public and private key. These keys are used during the SSL handshake [**?**] for mutual authentication and to establish a secure channel. This method is not secure unless a public key can be reliably mapped to an entity (a user or a service). GSI uses a third party called the Certifying Authority (CA) to certify this mapping.

Users and services generate public and private key pairs, and send the public keys to the CA for certification. The CA verifies - using some non-cryptographic means - that the public key belongs to that entity. Upon successful verification, it generates a document that contains the identity of the entity (called subject name), its public key, and the identity of the CA. This document is called a "certificate". This certificate is digitally signed by the CA, so that it cannot be tampered with.

During mutual authentication, both parties present their certificates to each other. For the authentication handshake to take place, the parties need to trust the Certifying Authorities of each other.

4.1.3 Proxies and delegation

In GSI, the private key of a user is stored on the user's machine. In order to protect it, it is never stored in its original form, but is encrypted using a passphrase provided by the user. Before it can be used, it needs to be decrypted. Thus, whenever a user wants to authenticate to a resource, he/she has to provide the passphrase in order to decrypt the private key. This can be very inconvenient, since a Grid computation typically involves obtaining access to many different computational and data resources. The need to enter the passphrase repeatedly can be avoided by creating what is called a "security proxy", hereafter referred to as "proxy".

A proxy contains a public and private key pair different than the key pair that belongs to the user. The proxy key pair is used during any authentication dialogs. A proxy has a limited lifetime, after which the keys are not valid. Thus, even if the private key gets compromised, the damage would be limited. This allows storing the private key of the proxy without encrypting it with a passphrase. Thus, there is no passphrase for the proxy.

A new certificate is generated for the proxy. It contains a mapping between the user's identity (slightly modified to denote that it is a proxy) and the new public key. This certificate is signed by the user, rather than a CA. The certificates thus form a trust chain, with the user's certificate signed by the trusted CA and the proxy certificate signed by the user.

In GSI, the long-term private key of the user cannot be used for authentication. It can only be used to sign the proxy, and that is the only time the user needs to enter his passphrase.

There are cases when a service needs to acquire resources on the behalf of the user. The user's proxy cannot be used for this, since it resides on the user's machine and not on the service machine. GSI uses a technique called "delegation" in such cases. When a user authenticates to a service and establishes an SSL connection, the user creates another proxy that is passed to the service. This proxy is signed by the private key of the user proxy, adding another link to the trust chain. The service can use this proxy to authenticate to other resources, on behalf of the user.

4.2 Security prerequisites

Most of the software provided by the Java CoG Kit uses the GSI security. To be able to start using GSI, you need to perform certain steps. This section describes these steps in detail.

4.2.1 Acquiring a user certificate

As mentioned in the introduction, getting a certificate for yourself is a matter of generating a public/private key pair, and sending it to the CA for identity verification and signing. The latter is site-specific. Each Grid should either have its own CA or use an established commercial CA that is trusted by the users and services in that Grid. Depending on the CA, the procedure for getting your certificate signed will vary.

Using Globus tools to acquire a user certificate

To use this method, you need to have an account on a machine where Globus is installed, and permission to run Globus tools. This method is described in detail on the following webpage:

Acquiring GSI certificates : `http://www.globus.org/security/v1.1/certs.html`

Follow the hyperlink "User certificates" on this page. Please note that the method specified on this page for sending your public key to the CA for signing is relevant only when using the Globus CA, and should be replaced by a procedure specific to your site. Please contact the Grid administrator at your site for details.

After you receive the certificate from the CA, please store the certificate and the private key file with appropriate file permissions in the `.globus` directory inside your home directory, as instructed on the above webpage.

4.2.2 Acquiring a host certificate (optional)

You need to acquire a host certificate if you are going to run a Globus service on your host. Examples of Globus services are Globus GRAM server (Section 6.1) , GridFTP server (Section 5.1.2), and so on. A host certificate is a binding between the identity of the host and its public key. Just as with user certificates, acquiring a host certificate is a matter of generating a public/private key pair, and sending it to the CA for identity verification and signing. Thus, it is site-specific, as explained in the previous subsection. Please note that *for acquiring a host certificate you need to have administrative privileges on the host.*

Using Globus tools to acquire a host certificate

Please refer to the following webpage:

Acquiring GSI certificates : `http://www.globus.org/security/v1.1/certs.html`

Follow the hyperlink "Host certificates" for instructions about acquiring a host certificate. As in the previous subsection, please replace the Globus-specific procedures with those specific to your site.

After you receive the certificate from the CA, please store the certificate and private key with appropriate file permissions in the host certificate directory, as mentioned on the above webpage.

4.2.3 Renewing a certificate

You will get a notification from the CA when your user or host certificate is about to expire. Renewing a new certificate involves generating a new public/private key pair, creating a renewal request and getting it signed from the CA. Thus, it is partly site-specific, as explained before. Globus provides a tool called `globus-cert-renew` for this. Please refer to the following webpage for a documentation of this tool.

Renewing a GSI Certificate : `http://www.globus.org/details/programs/globus-cert-renew.html`

4.2.4 Obtaining the certificates of the trusted CAs

You should only use a service if you trust the CA that signed the certificate. This essentially depends on whether you trust the administrators of the domain that hosts the service. If you decide to trust a particular CA, you need to obtain the certificate of that CA from its administrators. For example, administrators of the Globus CA make the certificate available for download at

Globus CA Certificate : `ftp://ftp.globus.org/pub/gsi/globus\protect\unhbox\voidb@x\kern.06em\vbox{\hrulewidth.3em}ca/42864e48.0`

.

Once you obtain the CA certificate, you need to let the Java CoG Kit know that you trust that CA. You can do this either by manually editing the `<user-home>/.globus/cog.properties` file or using the Java CoG Kit configuration wizard explained in Section 3.6.4

4.2.5 Gridmap files

Successful authentication alone is not sufficient for a user to use a service. Authentication only convinces the service that the user is indeed who he/she claims to be. In addition, the service has a right to check whether the user is authorized to use the service. It checks this on the basis of the user's Grid identity (i.e. his Distinguished Name), as found in the user's certificate.

Currently, Globus services perform authorization using a file called grid-mapfile that has to be present on every machine hosting a Globus service. This file is prepared by the GSI administrator of that site, depending on local policies. The file maps Grid identities to local usernames on that machine. A user is authorized to use a service only if the user's Grid identity can be mapped to his local username using the grid-mapfile.

Thus, before you can use any Globus service, you have to request the Grid administrator to add you to the grid-mapfile. The procedure is site-specific. Please contact the Grid administrator for details.

4.2.6 Protecting credentials

Please follow the steps given below in order to protect your security credentials:

- Make sure that the only permission on your long-term private key file `userkey.pem` is the read permission for yourself. This should be the case by default, and you should *never* change that.

- Make sure that the only permissions on your proxy file are read and write permissions for yourself. This should be the case by default, and you should *never* change that.
- If you are running a Globus service, make sure that the only permission on the long-term private key of the host `hostkey.pem` is the read permission for the superuser (root/administrator). This should be the case by default, and you should *never* change that.
- In the process of acquiring user credentials, you are prompted to enter a passphrase. This passphrase is used to encrypt your long-term private key. Please make sure to select a passphrase that is easy to remember for you, but still very difficult to guess for others. If, by mistake, you have chosen a weak passphrase during this process, please change the passphrase using the tools described in Section 4.4.2 or 4.4.3.
- Please don't store the private key and proxy files mentioned above on removable media like floppy disks or zip disks, which may get stolen.
- If you want to copy the private key and proxy files mentioned above to some other host, please don't use insecure methods like FTP or rcp.

4.3 MyProxy

Users may use different computers to access services on the Grid. These may include computers at work, computers at home, and public access terminals. All of these machines may not be very secure and trustworthy. On each of these machines, the user needs to have his security credentials in order to authenticate to the Grid services. But it is not secure to copy the long-term credentials (user's private key and certificate) to every machine, as they may be compromised. Instead, it is desirable to have a central secure server trusted by the user where the user can store his credentials and later retrieve a proxy whenever needed for authentication. Since proxies have a limited lifetime that can be controlled, the compromise of a proxy does not cause much damage. MyProxy [3] serves this purpose. A securely managed MyProxy server that is trusted by the user can provide an effective way of credential management. MyProxy is available from the following website:

MyProxy Homepage : `http://www.ncsa.uiuc.edu/Divisions/ACES/MyProxy/`

First, the user has to store a credential to the MyProxy server. This is done from a machine that has the user's long-term credentials. From these credentials, a proxy is created and sent to the MyProxy server. The lifetime of the proxy on the MyProxy server can be controlled by the user. The proxy can be secured using a username and a password. Also, the user can restrict the hosts which can later retrieve and/or renew the proxy.

At a later time, a proxy can be retrieved by supplying the username and password that the user has set. The lifetime of the retrieved proxy can be controlled. The proxy can also be renewed, if needed.

Grid administrators may refer to the Administrator's Guide on the MyProxy homepage mentioned above, for instructions on how to maintain a MyProxy server. For users, MyProxy software comes with tools to store and retrieve credentials. The Java CoG Kit also provides command line tools for this purpose. Please refer to sections 4.4.2 and 4.4.3 for information about using these tools.

4.4 Managing certificates and proxies

Some of the tools described in this section need that the environment variable `COG_INSTALL_PATH` is set to `<cog-install-path>`, as discussed in Section 3.6.1.

4.4.1 GUI

Currently, the Java CoG Kit provides the following GUI-based tools for credential management:

Visual-grid-proxy-init

This tool allows creation of a proxy. Lifetime and cryptographic strength of the proxy can be specified. Also, the locations of user's long-term credentials and the location of the resulting proxy file can be specified.

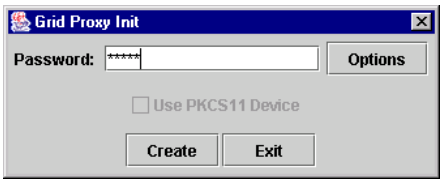

Figure 4.1: Visual-grid-proxy-init.

To run this tool, run the shell script `visual-grid-proxy-init` or the Windows batch file `visual-grid-proxy-init.bat` in the `<cog-install-path>/bin` directory.

Java CoG Kit configuration wizard

This tool lets the user configure the Java CoG Kit by specifying various security related parameters, such as the locations of the user's long-term and proxy credentials, locations of the files containing trusted CA certificates, and some other options. The tool then creates a configuration file called cog.properties, which is used by the Java CoG Kit software. This tool is described in detail, along with screenshots, in Section 3.6.4.

To run this tool, run the shell script `ogce-setup` or the Windows batch file `ogce-setup.bat` in the `<cog-install-path>/bin` directory.

4.4.2 Unix shell scripts

The Java CoG Kit provides a number of Unix command line tools. All of these tools can be found in the `<cog-install-path>/bin` directory. Each of these tools supports a `-help` command line option that prints a detailed usage message describing various options. These usage messages have also been included in the Appendix A of this manual.

grid-proxy-init

Allows creation of a proxy. By default, this tool generates a GSI-3 style proxy. The GSI-3 style proxies are not compatible with older servers (such as Globus Toolkit 2.2, 2.0). Thus, they will not work for majority of the examples in this manual.

To generate a proxy that is compatible with Globus Toolkit 2.2 and 2.0 servers, either use the `visual-grid-proxy-init` tool described in the previous section, or use the `-old` option for this tool. Other options supported by this tool are lifetime, strength, policy file, etc. This tool prompts you for your passphrase. Usage syntax is

```
> grid-proxy-init [options]
```

For example, to create a proxy that will work with Globus Toolkit 2.2 and 2.0 servers, has a validity period of 12 hours, and contains 1024 bit keys,

```
> grid-proxy-init -old -hours 12 -bits 1024
```

Warning: The grid-proxy-init tool echoes your passphrase to the screen. The reason for this is currently Java does not have a portable way of reading the passphrase securely from the console (without echoing it to the screen first) Any solution would sacrifice the portability of the code. If you want to avoid this behavior, please use the visual-grid-proxy-init tool described in the previous section.

grid-proxy-info

Displays information regarding a proxy. It can display various pieces of information, such as the issuer, Distinguished Name (DN) of the identity, time left on the proxy and so on. Usage syntax is

```
> grid-proxy-info [options]
```

For example, to observe the identity and validity period of a proxy, use

```
> grid-proxy-info -identity -timeleft
```

grid-proxy-destroy

Destroys a user proxy, if present. The files containing proxies can be specified. The `dryrun` option prints the names of files that would be destroyed, without actually destroying them. Usage syntax is

```
> grid-proxy-destroy [-dryrun] [file1, file2, ...]
```

grid-cert-info

Displays information regarding the long-term user certificate. It can display the identity that the certificate represents, the CA that signed the certificate, the validity period, and so on. Usage syntax is

```
> grid-cert-info [options]
```

grid-change-pass-phrase

Allows changing of the passphrase used to encrypt the long-term private key of the user. You need to enter your old passphrase. Usage syntax is:

```
> grid-change-pass-phrase [options]
```

Warning: The grid-change-pass-phrase tool echoes your old and new passphrases to the screen. The reason for this is currently Java does not have a portable way of reading a passphrase securely from the console (without echoing it to the screen first) Any solution would sacrifice the portability of the code. The only way to avoid this problem is to provide a graphical front-end to grid-change-pass-phrase. Though this is not currently available, we plan to develop this in future.

myproxy

Allows storing and retrieval of credentials using a MyProxy server. Supports various options like the hostname and port number of the server, the lifetime of the delegated proxy, etc. Usage syntax is

```
> myproxy [options] command
```

where command is one of `put`, `get`, `anonget`, `destroy`, and `info`. `anonget` is used for anonymous retrievals.

For example, to store a proxy to host `myproxy.mcs.anl.gov`, with a validity period of 12 hours, use

```
> myproxy -h myproxy.mcs.anl.gov -c 12 put
```

In case of the `put` command, you will be prompted first for your Grid passphrase, and then for the password to be used to protect the credential stored on the MyProxy server. When you later try to retrieve the credential using `get` or `anonget` or any other method, you will be asked to enter this password.

Warning: The myproxy program echoes both your Grid passphrase and the credential password to the screen.

The reason is same as the one mentioned above regarding grid-change-pass-phrase. This problem will go away when we provide a graphical front-end to this tool.

4.4.3 Windows batch files

Each of the tools described in the previous section has a Windows batch file counterpart. These batch files can be found in the `<cog-install-path>/bin` directory. Just like the Unix shell scripts, each of them supports a `-help` option that prints a usage message. The usage details have also been included in the Appendix A for this manual.

4.4.4 Java CoG Kit shell

The Java CoG Kit Shell is a convenience application that allows you to use several Java CoG Kit features from a platform independent command line interface.

To start the shell, execute the following from the `<cog-install-path>/bin` directory:

```
> cog-shell
```

proxy-init

Currently, the Java CoG Kit shell provides a single command called `proxy-init`, that presents a GUI to create a user proxy. This GUI is the same as the one explained in Section 4.4.1.

4.4.5 API

In this section we describe selected security APIs provided by the Java CoG Kit. Please note that these APIs are different from (and not backwards-compatible with) the APIs in the previous versions of the Java CoG Kit. For the convenience of developers used to the old APIs, we also provide a comparison of the old and new library APIs in this section.

Reasons for changing the security library API.

1. The old security library was based on a commercial SSL library (IAIK), which had licensing restrictions not suitable for many of the Java CoG Kit users.
2. The old security library was socket-oriented (it was difficult to write non-socket based security modules e.g. for FTP, MDS, etc.)
3. The old security library API was not designed to work with multiple security protocols, represent different types of credentials, etc.

Functionality provided by the new library

The new security library is based on GSS-API and is implemented entirely with open-source SSL and certificate processing libraries. With the GSS-API abstractions it is possible to provide transport and security protocol independence. Also, the new library supports a few new features such as the new proxy certificate format and delegation-at-any-time API. For a detailed list of GSS-API implementation features and limitations, please see the following webpage:

Java GSI GSS-API Implementation : http://www.globus.org/cog/distribution/1.1/api/org/globus/gsi/gssapi/Java_GSI_GSSAPI.html

Key differences between old and new library

1. GSS abstractions are used throughout the code instead of the old security API (e.g. previously, setCredential(org.globus.security.GlobusProxy) and now setCredential(org.ietf.jgss.GSSCredential))
2. All the security classes in the org.globus.security package and all subpackages (except org.globus.security.gridmap package) are now deprecated.

3. The functionality of the org.globus.security.GlobusProxy class is mostly replaced by org.globus.gsi.GlobusCredential class. However, it is strongly recommended not to use (if possible) org.globus.gsi.GlobusCredential class as it is security-protocol specific representation of (PKI) credentials. Instead, it is recommended to use the GSS abstractions as much as possible as shown in the sample code in this section.

Getting default (user proxy) credentials

Versions of Java CoG Kit before 1.1a

```
GlobusProxy cred = GlobusProxy.getDefaultUserProxy();
```

Java CoG Kit 1.1a

```
ExtendedGSSManager manager = (ExtendedGSSManager)
                              ExtendedGSSManager.getInstance();
GSSCredential cred = manager.createCredential(
                             GSSCredential.INITIATE_AND_ACCEPT);
```

Saving credentials in a file

Versions of Java CoG Kit before 1.1a

```
GlobusProxy cred = ...
FileOutputStream out = new FileOutputStream("file");
cred.save(out);
out.close();
```

Java CoG Kit 1.1a

```
ExtendedGSSCredential cred = ...
byte[] data = cred.export(ExtendedGSSCredential.IMPEXP_OPAQUE);
FileOutputStream out = new FileOutputStream("file");
out.write(data);
out.close();
```

Loading credentials from a file

Versions of Java CoG Kit before 1.1a

```
FileInputStream in = new FileInputStream("file");
GlobusProxy cred = GlobusProxy.load(in, null);
in.close();
```

Java CoG Kit 1.1a

```
byte [] data = new buffer[1024];
FileInputStream in = new FileInputStream("file");
// read in the credential data
in.read(data);
```

```
in.close();
ExtendedGSSManager manager = (ExtendedGSSManager)
                              ExtendedGSSManager.getInstance();
GSSCredential cred = manager.createCredential(data,
                          ExtendedGSSCredential.IMPEXP_OPAQUE,
                          GSSCredential.DEFAULT_LIFETIME,
                          null, // use default mechanism - GSI
                          GSSCredential.ACCEPT_ONLY);
```

Getting the remaining lifetime of a credential

Versions of Java CoG Kit before 1.1a

```
GlobusProxy cred = ...
int time = cred.getTimeLeft();
```

Java CoG Kit 1.1a

```
GSSCredential cred = ...
int time = cred.getRemainingLifetime();
```

Getting the identity of the credential

Versions of Java CoG Kit before 1.1a

```
GlobusProxy cred = ...
String identity = CertUtil.toGlobusID(cred.getSubject());
```

Java CoG Kit 1.1a

```
GSSCredential cred = ...
String identity = cred.getName().toString();
```

GlobusCredential/GSSCredential conversion

As mentioned before, it is not recommended to use the GlobusCredential class directly. To convert an instance of GlobusCredential to a GSSCredential instance, you must first wrap it in org.globus.gsi.gssapi.GlobusGSSCredentialImpl class, as shown below:

```
GlobusCredential cred = ...
GSSCredential cred = new GlobusGSSCredentialImpl(cred,
                               GSSCredential.ACCEPT_ONLY);
```

It is also possible to retrieve the org.globus.gsi.GlobusCredential object from the GSSCredential instance if it is of the right type:

```
GSSCredential cred = ...
if (GSSCredential instanceof GlobusGSSCredentialImpl)
{
  GlobusCredential globusCred =
```

```
            ((GlobusGSSCredentialImpl)cred).getGlobusCredential();
    }
```

4.5 Firewall Issues

Grids usually involve multiple organizations, or at least multiple departments in the same organization. Thus, the interactions between Grid clients and servers span network boundaries. Some of these networks may have firewalls. Since the network communication in Globus is based on TCP sockets, firewalls may block it. You may face the following issues due to network firewalls, while using the Java CoG Kit to communicate with Globus servers.

- Connecting to Globus servers: If the port a Globus server is listening on is blocked by the firewall, the connection will fail. This applies to all servers like GridFTP servers, Globus Gatekeepers, and so on. This problem can be solved if the person maintaining the server requests the administrators of the server network to configure the firewall such that it allows traffic destined for Globus servers. Since most of the Globus servers listen on fixed, well-known ports, this is possible. Please refer to the following webpage for a list of common Globus servers and the ports they listen on.

Globus Toolkit firewall requirements : http://www.globus.org/security/v1.1/firewalls.html

- GridFTP data channels: While transferring data using GridFTP (introduced in Section 5.1.2), if the client (your computer) is set in Passive mode, it starts listening on an available port, and conveys this port number to the server. The server will then try connecting to that port. Since this port is neither fixed nor well-known, a firewall on the client's network will probably block it. Thus, the server will not be able to connect to the port. The solution is to enforce the client to listen on a port number that lies in a specific range of ports, and then request the network personnel to allow these ports through the firewall. The methods used to restrict the client to a specific port range are described later in this section.
- GASS servers setup for file staging and output/error retrieval: As explained in Section 6.3.5, a GASS server needs to be run on a client machine (your machine) for staging executables and data files to, and retrieving output/errors from jobs running on a remote GRAM server. The GRAM Job Manager will try to establish a connection with the GASS server on the client machine. Since the ports used by GASS servers are neither fixed nor well known, a firewall in the client's network will probably block the connection. The solution is the same as the one mentioned above for GridFTP data channels: restricting the port range.
- Connecting to Globus servers behind a NAT firewall: Some networks employ a firewall that performs Network Address Translations for the hosts in those networks. Please refer to the following webpage for a discussion of problems posed by NAT firewalls and possible solutions to these problems.

Globus Toolkit firewall requirements : http://www.globus.org/security/v1.1/firewalls.html

Restricting the port range used by GridFTP data channels and GASS servers

The port range can be set either in the Java CoG Kit configuration file, i.e. `<user-home>/.globus/cog.properties`, or through the Java system properties, e.g. set from the command line. To set the port range in the configuration file just add the following line to the file:

```
tcp.port.range=<min>,<max>
```

For example,

```
tcp.port.range=6000,6060
```

To set the port range using the system properties, set the `org.globus.tcp.port.range` property.

For example:

```
> java -Dorg.globus.tcp.port.range="6000,6060" -classpath ...
```

4.6 Random number generation issues

For security-related tasks, the Java CoG Kit tools and APIs must initialize a secure seed for the random number generator. On some platforms this may be a very computationally expensive process. However, the seed for the random number generator only needs to be initialized once per Java Virtual Machine instance. The Java CoG Kit can be configured to use an arbitrary SecureRandom implementation (which can be optimized for particular platform(s)) by adding the following properties into the `<user-home>/.globus/cog.properties` file:

```
random.provider=<Provider class>
random.algorithm=<algorithm name>
```

For example, if you are using the ISNetworks implementation of SecureRandom, add the following into the `<user-home>/.globus/cog.properties` file:

```
random.provider=com.isnetworks.provider.random.InfiniteMonkeyProvider
random.algorithm=InfiniteMonkey
```

SecureRandom implementation by ISNetworks : http://www.isnetworks.com/infinitemonkey/

Of course, you must first install the provider correctly.

The next time you use a Java CoG Kit tool or library the startup time should be faster on the platforms supported by the provider. For platforms not supported by the provider, the default seed generator will be used.

5 File I/O and Transfer

This chapter begins with a discussion of file access and transfer issues in a Grid environment. It then introduces various methods of data access and transfer over the Grid. It gives an overview of the GridFTP and GASS protocols. It then describes various file transfer tools and APIs provided by the Java CoG Kit. Some example code demonstrating the use of file transfer APIs is provided.

5.1 Introduction

An important aspect of distributed computing is access to distributed data. Many Grid-based scientific and engineering applications require transfer of large amounts of data (terabytes or petabytes) between storage systems, and access to this data by applications running on remote hosts. For example, data generated by a particle accelerator for a multinational physics collaboration may need to be transferred to analysis centers in different continents, from where the data would be accessed by multiple analysis applications.

5.1.1 Requirements for File Access and Transfer over the Grid

Among the most important requirements are high performance, security, reliability and restartability. A few more requirements are imposed by the high heterogeneity involved in the Grid environment. The Grid being a multi-organizational environment, different storage systems, operating systems, security infrastructures, resource namespaces, etc. are used at different sites. It is clearly inconvenient for the applications to use different protocols and APIs to interact with different systems. The file access and transfer mechanism chosen for the Grid should, therefore, provide an abstract layer for access, irrespective of the underlying heterogeneity. At the same time, it should impose as few requirements as possible on the resource providers, to make it easy for them to incorporate their resources into the Grid environment.

The Globus Toolkit provides two methods for accessing distributed data. As a data transfer protocol, it provides Grid File Transfer Protocol (GridFTP) [6, 7, 8], which is a common protocol independent of the underlying architectures. It supports GSI and Kerberos security. It also provides various features for high performance, reliable and restartable data transfers, as mentioned in the next section. The other method, Globus Access to Secondary Storage (GASS) [9, 10], allows applications to use standard file I/O interfaces (open, read/write, close) for distributed access. It defines a global name space using Uniform Resource Locators. It also allows the use of GSI security for file access. Thus, it makes porting of applications to the Globus environment easy.

5.1.2 GridFTP

GridFTP is a set of extensions to the FTP protocol that provide increased security, reliability and performance to data transfers.

GridFTP Protocol Specification : http://www.globus.org/research/papers/GridftpSpec02.doc

FTP was chosen as the basis because of its widespread use, easy extensibility, separation of control and data channels, and so on. GridFTP protocol [6] provides features such as GSI security for both control and data channels, parallel transfers, striped transfers, partial file transfers, and third-party transfers. In parallel transfers, data from a single file can be split over multiple data connections. A striped transfer distributes data in a file over multiple independent data nodes. A third party transfer takes place between servers A and B, while the client C manages the transfer. GridFTP allows monitoring the progress of a transfer using "performance markers", which are essentially progress indicators sent by the server periodically. GridFTP servers may also send "restart markers", which act as checkpoints for the transfer. If a transfer fails at any time, it can be resumed from the last checkpoint. The bytes already transferred before the last checkpoint do not have to be transferred again.

GridFTP is typically made available as a standard service on a server running the Globus Toolkit. By default, it listens on port 2811.

5.1.3 GASS

GASS is a mechanism to read and write remote files using secure HTTP protocol. GASS clients developed in C provide applications with special functions to open and close remote files. After this, applications can use the normal C library read() and write() functions. Since Java uses the concept of streams for I/O, Java CoG Kit client APIs provide input and output streams to access remote files.

GASS can also be used by Globus GRAM servers to transfer executables and data files needed for a computational job, from any host on the Grid. This method, called file staging, frees the user from transferring the file manually. Similarly, job output and error files can be sent to any host on the Grid. Both file staging and output/errors redirection need a GASS server to be run on the machine submitting the job. Please refer to Section 6.3.5 for more information about this. A single GASS server can be used to stage files and receive output/errors for multiple jobs running on multiple remote sites.

Unlike a GridFTP server, a GASS server presently cannot serve files to multiple users. Thus, it is not possible to have just a single GASS server running as root on a host. Instead, a user wishing to access files using GASS has to start his own GASS server, which is basically a HTTPS server. On the local machine, this can be done simply by creating a HTTPS server process. For starting up a GASS server on a remote host, however, the Globus gatekeeper has to be used. The Java CoG Kit provides tools and APIs for starting local and remote GASS servers.

GASS cache

To increase performance of file access, GASS supports the concept of a "file cache" on a host running the Globus GRAM service. As mentioned before, executable and data files needed by jobs can be staged from different hosts. If multiple jobs access the same remote file, it can be cached for better performance. The Globus Toolkit also allows the users to add, delete and list files in the GASS cache, with a command line utility called `globus-gass-cache`. Adding a file could be useful, for example, to stage files before job execution starts, in order to avoid any delay in processing.

Currently, Java CoG Kit does not provide support for GASS cache. Please see Section 5.3.6 for more details.

5.1.4 Other file transfer mechanisms

In addition to the specific mechanisms discussed above, methods like regular FTP and Secure Copy (scp) can also be used for file transfers over the Grid, though they may not satisfy the different requirements discussed in Section 5.1.1.

The Java CoG Kit provides client APIs for FTP transfers. For these transfers, you have to use the username/password authentication method. You cannot use the GSI authentication. Also, these transfers cannot use the features provided exclusively by GridFTP, as mentioned in Section 5.1.2. Furthermore, the use of FTP over untrusted networks is discouraged, because it sends passwords across the network in cleartext.

Secure copy (scp), which is based on SSH, does not have the problem of cleartext passwords. It may suffer, however, from man-in-the-middle attacks due to the lack of certificates. Toolkits like dsniff [11] demonstrate this vulnerability.

We do not discuss these mechanisms in more detail, as they are beyond the scope of this document. Instead, we concentrate on GridFTP and GASS.

5.1.5 Security Requirements

Both GridFTP and GASS use the GSI for authentication and secure data transfer. Thus, you will need to acquire the GSI credentials before you can transfer any data. Please refer to the Sections 4.2.1, 4.2.4 and 4.2.5 for instructions about acquiring the required credentials.

5.2 Using GridFTP

Some of the tools described in this section need that the environment variable `COG_INSTALL_PATH` is set to `<cog-install-path>`, as discussed in Section 3.6.1.

5.2.1 GUI

The Java CoG Kit provides a tool called File Transfer GUI with an easy-to-use interface for connecting to multiple FTP and GridFTP servers and transferring files. You can also browse the local file system with this tool. File and directory transfers between the local system and remote systems, and between two remote systems (direct third-party transfers) are provided. File System operations such as creating, deleting and renaming files and directories are also provided.

This tool provides both client and server side reliability. The client side reliability is provided by the tool itself using the Java CoG Kit File Transfer Service. The server side reliability is provided by interfacing to the Globus Toolkit 3 OGSA-based Reliable File Transfer (RFT) service [12] developed at Argonne National Laboratory.

Client Side Reliability

At the client side, the tool allows the user to make directory transfer requests, store them in a queue and monitor the transfers. If there is any failure during the transfer

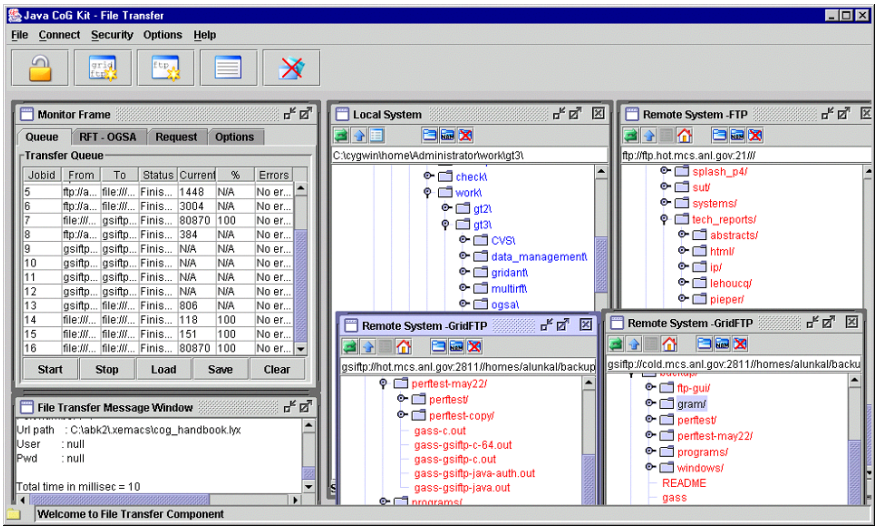

Figure 5.1: File Transfer GUI.

due to network outrage, the tool alerts the user and continues the transfer after recovering from the failure. It allows the user to save transfer requests in a file and make the transfers at a later time.

Server Side Reliability

The setting of RFT requires few additional steps during the setup, which is explained in the next paragraph. Given the source and the destination of the transfer, this service performs the transfer reliably, recovering automatically from certain types of failures such as server crashes and network outages. The service after forking off the transfer client monitors the transfer by waiting on the transfer client.If the client returns a fatal error (e.g when the source URL or destination URLs are not valid among other things) which means the transfer is impossible to do then the service will not restart the failed transfer but if the client returns a non fatal error (which can be anything from a crashed server to network outage) the service will restart the transfer. The transfer is started from the point where it failed before.

If you want to use the RFT feature, you need to build the GT3 RFT client. The following steps are needed:

1. Get the source code distribution of the Java CoG Kit and compile it as described in Section 3.4.5.
2. You need to check out the gridant module from the cvs repository into the same directory where ogce and jglobus are checked out.

   ```
   > cvs -d :pserver:anonymous@cvs.globus.org:/home/dsl/cog/CVS co gridant
   ```

3. Build the gt3 RFT client by using the following command:

   ```
   > cd ogce
   > ant gt3
   ```

   All the jar files needed to interface to the RFT server are copied into the destination directory.

4. You need to run GT3 Reliable File Transfer service by following the instructions at [12].

The setup is complete. You can run the tool using ant as follows:

```
> cd ogce
> ant -f demos.xml ftp
```

You can also run the shell script `cog-ftp` or the Windows batch file `cog-ftp.bat` in the `<cog-install-path>/bin` directory.

Inorder to interface to the RFT service, you need to edit the following options:

1. Edit the server Textfield available in the Options Tab RFT section in the GUI Tool to specify the location of the Reliable File Transfer service.
2. Select the Remote GT3 Provider in the Options Tab.

When you drag and drop the directories or files, the requests are send to the remote Reliable File Transfer Service which does the actual transfer. If the RFT service is not setup, then the tool uses the transfer service provided by the Java CoG Kit.

5.2.2 Unix Shell Scripts

The tools described in this section can be found in the `<cog-install-path>/bin` directory.

globus-url-copy

Allows file transfers between a local system and a remote system, or between two remote systems. The source and target locations are specified as URLs. The usage syntax is as follows:

```
> globus-url-copy options fromURL toURL
```

A complete list of available options can be obtained by running

```
> globus-url-copy -help
```

This list has also been included in the Appendix A.

`globus-url-copy` supports FTP and GridFTP protocols. It also supports HTTP and HTTPS protocols for GASS transfers. The protocol-specific URL formats are

- FTP: `ftp://<user>:<password>@<host>:<port>/<file-path>`
- GridFTP: `gridftp://<host>:<port>/<file-path>`
- HTTP: `http://<host>:<port>/<file-path>`
- HTTPS: `https://<host>:<port>/<file-path>`
- Local files: `file:///<file-path>` (please note the three slashes)

Notes:

1. `<port>` is optional in all cases.

2. In case of FTP, username and password should both be provided or omitted. In case they are omitted, an anonymous connection will be made.
3. If the HTTP(S) URL is referring to a GASS server running on a Unix-like operating system, `<file-path>` would be a hierarchical path relative to the root ("/") directory. It will look like `<directory>/<directory>/<directory>/.../<name>`.
   For example, home/albert/document.tex
4. If the HTTP(S) URL is referring to a GASS server running on Windows, `<file-path>` would be of the form `<drive-letter>:/<directory>/<directory>/.../<name>`.
   For example, c:/temp/myfolder/document.txt

For example, to transfer a file from an anonymous FTP server to a GASS server running on a Windows machine ,

```
> globus-url-copy ftp://ftp.foo.org/banner.msg
                  https://hot.anl.gov:2222/c:/temp/banner.msg
```

5.2.3 Windows Batch Files

`globus-url-copy`, explained in the previous section, has a Windows batch file counterpart named `globus-url-copy.bat`. It can be found in the same location as `globus-url-copy`. The capabilities are identical.

5.2.4 Java CoG Kit Shell

The Java CoG Kit Shell is a convenience application that allows you to use several Java CoG Kit features from a platform independent command line interface. Currently, an equivalent of `globus-url-copy` is under development for this shell. Text-based interactive interfaces for FTP and GridFTP are also in progress. These will be similar to the "ftp" program available on many Unix systems. To start the Java CoG Kit shell, execute the following from the `<cog-install-path>/bin` directory:

```
> cog-shell
```

5.2.5 APIs

The Java CoG Kit provides a set of APIs for file transfers using FTP and GridFTP. We show here some examples that use the GridFTP APIs. For a detailed programmer's guide and complete documentation in Javadocs format, please refer to the following website:

Java CoG Kit File Transfer API guide : http://www.globus.org/cog/jftp/

Specifically, the programmer's guide addresses the following issues:

- File storage and retrieval to and from FTP and GridFTP servers
- Third-party (direct server-to-server) transfers between FTP and GridFTP servers
- Parallel and Striped transfers using GridFTP

- Measuring performance of a file transfer
- Restarting failed transfers

Transferring files between a client and a server

As described before, GridFTP uses the GSI security mechanism. Thus, APIs described here need security credentials in the form of an object of the class org.ietf.jgss.GSSCredential. Please refer to the Section 4.4.5 for details about how you can use the Java CoG Kit APIs for getting GSSCredential objects from your GSI proxies.

```
/**
 * Get a GSSCredential object as explained above.
 */

GSSCredential credential;

/**
 * Create an instance of the GridFTPClient class.
 */

String host = "hot.mcs.anl.gov";
int port = 2811;
GridFTPClient hotClient = new GridFTPClient(host, port);

/**
 * Authenticate to the server
 */

hotClient.authenticate(credential);

/**
 * Set security parameters such as data channel authentication
 * (defined by the GridFTP protocol) and data channel
 * protection (defined by RFC 2228).

 * If you do not specify these, data channels are authenticated
 * by default.
 */

hotClient.setProtectionBufferSize(16384);
hotClient.setDataChannelAuthentication(DataChannelAuthentication.SELF);
hotClient.setDataChannelProtection(GridFTPSession.PROTECTION_SAFE);

/**
 * Get a list of files and directories in the current directory.
 * The function returns a vector of FileInfo objects.
 * Each of these objects contains information about
 * a remote file, such as name, size, modification time, etc.
 */

Vector fileInfoVector = hotClient.list();

/**
 * Get a file from the remote server.
 */

String remoteFile1 = "testDir/getFile.txt";
File localFile1 = new File("getFile.txt");
hotClient.get(remoteFile1, localFile1);

/**
 * Send a file to the remote server.
```

```
 */

boolean append = true;
String remoteFile2 = "testDir/putFile.txt";
File localFile2 = new File("putFile.txt");
hotClient.put(localFile2, remoteFile2, append);
```

Third-party transfers

Following is an example showing a third-party transfer between two GridFTP servers, namely, hot.mcs.anl.gov and cold.mcs.anl.gov. The former is assumed to be the source of the file and the latter is assumed to be the destination.

```
/**
 * Create a GridFTPClient object for cold.mcs.anl.gov,
 * and perform authentication.
 */

GridFTPClient coldClient =
             new GridFTPClient("cold.mcs.anl.gov", 2811);
coldClient.authenticate(credential);

/**
 * Set the data channel authentication and protection
 * parameters, as shown above for hotClient.
 */

/**
 * The following step is optional, unless using Extended
 * Block Mode. It is performed here for illustrative purposes.

 * Set the receiving server to passive mode, so that
 * it starts listening for a data channel connection, on any available port.
 * Set the sending server to active mode, providing it with
 * the above-mentioned port and the hostname of the receiving server,
 * so that the sending server can open a data channel connection to
 * the receiving server.
 * These operations, if performed, have to be in that order.
 */

HostPort hp = coldClient.setPassive();
hotClient.setActive(hp);

/**
 * Transfer a file.
 * The transfer() function blocks until the transfer
 * is complete.
 */

String remoteSrcFile = "testDir/srcFile";
String remoteDstFile = "testDir/dstFile";
append = true;
hotClient.transfer(remoteSrcFile,
                coldClient, remoteDstFile,
                append, null);
/*
 * Close both the servers. This is very important,
 * as it releases the resources and saves you from
 * running out of memory, as explained in the GridFTP
 * Programmer's Guide.
 */

hotClient.close();
coldClient.close();
```

A full-fledged, running example is available at the following location:

ogce : `org/globus/examples/HelloGridFTP.java`

5.2.6 Differences between Java CoG Kit version 0.9.13 and 1.1a

Java CoG Kit 0.9.13, when initially released, only provided the library org.globus.io.ftp. Later, Jftp was released, containing package org.globus.ftp. This package was the new implementation of the GridFTP protocol, and was compatible with the Java CoG Kit 0.9.13. The two packages co-existed for some time, but the use of the package org.globus.io.ftp was discouraged. Now that package has been removed from the distribution, and is no longer supported. Users should use the org.globus.ftp package. A list of the FTP and GridFTP protocol features supported by the latter is provided in the next section.

5.2.7 FTP/GridFTP protocol features supported by the Java CoG Kit

The following is a list of all protocol features of FTP and GridFTP that are supported by the Java CoG Kit.

FTP

- file storage and retrieval to/from FTP server (client-server transfer)
- third party transfer
- data channel protection level [13] (clear, safe, private)
- ASCII and IMAGE data types
- file data structure
- non print format control
- stream transmission mode
- operation in passive and active server mode

GridFTP 1.0 (in addition to the aforementioned)

- Mode E
- parallel transfers
- striped transfers
- IMAGE data type, if in mode E
- restart markers
- performance markers
- data channel authentication
- SBUF setting TCP buffer size

5.2.8 Limitations of the Java CoG Kit

Unsupported features of GridFTP

Following are the GridFTP 1.0 features not provided by the Java CoG Kit.

- ABUF
- PIPE (pipelining of commands)
- partial file transfer
- any combination of transfer parameters that is not mentioned above, for instance: mode E with ASCII

If you need any of these features, please send a request using the Bugzilla system, as explained in Section 2.2.2. Please be sure to include a brief description of your project, and how the particular feature may help the project.

Support for limited directory listing formats

The output of the list function in FTP servers depends on the particular FTP server, operating system and the architecture of the machine that the server is running on. Even the same FTP server software running on various Unix platforms may produce different results. Any non-Unix FTP server may produce a completely different representation. The FTP library in the Java CoG Kit is designed to handle the following Unix-like file list formats:

```
-rw-r--r-- 1 gawor globus 528 Nov 23 15:10 Makefile
```

and

```
-rw-rw-r-- 1 globus 117579 Nov 29 13:24 AdGriP.pdf
```

Any other file list format will not be parsed and an exception will be returned to the user. If you are using the API, you can write your own parser for the particular format you are interested in. For this, you have to use the parameterized `list(...)` function in `FTPClient` or `GridFTPClient` class, and intercept the input to the `DataSink` interface. Please refer to the Javadocs for more details.

5.3 Using GASS

Some of the tools described in this section need that the environment variable `COG_INSTALL_PATH` is set to `<cog-install-path>`, as discussed in Section 3.6.1.

5.3.1 GUI

Currently, the Java CoG Kit does not provide any GUI tools for using GASS.

5.3.2 Unix Shell Scripts

Java CoG Kit provides a number of Unix command line tools. All of these tools can be found in the `<cog-install-path>/bin` directory. Each of these tools supports a `-help` command line option that prints a detailed usage message describing various options. These usage messages have also been included in the Appendix A of this manual.

globus-gass-server

Starts a GASS server on the local machine, and prints its URL. Port number may be specified. You can control the level of access this server will have to the local file system. Access can be read-only or write-only or read/write. Redirection of standard output and error streams of a job can be controlled. You can also specify whether this server can be shut down with a request from a client. Usage syntax is

```
> globus-gass-server [options]
```

For example, to start a server listening on port 2222, with read-only access to local file system:

```
> globus-gass-server -p 2222 -read
```

globus-gass-server-shutdown

Stops a GASS server, given its URL. For this to succeed, the server must allow client-initiated shutdowns. The GASS server can be local or remote. Usage syntax is:

```
> globus-gass-server-shutdown [options] <GASS-URL>
```

For example, to shut down a GASS server running on hot.mcs.anl.gov on port 2345,

```
> globus-gass-server-shutdown https://hot.mcs.anl.gov:2345/
```

globus-url-copy

Please refer to Section 5.2.2 for a description of `globus-url-copy`.

5.3.3 Windows Batch Files

Each of the tools described in the previous section has a Windows batch file counterpart. These batch files can be found in the `<cog-install-path>/bin` directory. Just like the Unix shell scripts, each of them supports a `-help` option that prints a usage message. The usage details have also been included in the Appendix A for this manual.

5.3.4 Java CoG Kit Shell

The Java CoG Kit Shell is a convenience application that allows you to use several Java CoG Kit features from a platform independent command line interface. Currently, an equivalent of `globus-url-copy` is under development for this shell. To start the shell, execute the following from the `<cog-install-path>/bin` directory:

```
> cog-shell
```

5.3.5 APIs

This section discusses the Java CoG Kit APIs for GASS. Many of the APIs described here need security credentials in the form of an object of the class org.ietf.jgss.GSSCredential. Please refer to the Section 4.4.5 for details about how you can use the Java CoG Kit APIs for getting GSSCredential objects from your GSI proxies. In the following sections we assume that you have already created objects of the GSSCredential class.

Starting a local GASS server

The procedure to start a local GASS server is described in detail in Section 8.3.2. As mentioned before, a GASS server is started on the local machine mainly for staging files and receiving output/errors from jobs submitted to remote Globus GRAM servers.

Starting a remote GASS server

```
/**
 * Create a RemoteGassServer instance

 * cred - the security credential (identity certificate,
 * private key) the server will use for authentication
 * Object of class org.ietf.jgss.GSSCredential.

 * port - the port on which the server should listen
       if 0, a dynamic port will be assigned
 */

boolean secure = true;

RemoteGassServer server = new RemoteGassServer(cred,
                                      secure, port);

/**
 * Set the options for read/write access to remote file
 * system, output/error redirection, and for client-initiated
 * shutdowns.

 * options - a bitwise OR of zero or more of the following
 * flags:
 * READ_ENABLE, WRITE_ENABLE, STDOUT_ENABLE, STDERR_ENABLE,
 * CLIENT_SHUTDOWN_ENABLE.
 * These flags are static variables defined in the class
 * org.globus.io.gass.server.GassServer:

 * For example, if you want to allow client-initiated shutdowns
 * and read-only access to remote file system, use a bitwise
 * OR of those two flags, as shown in the code below.
 */

int options = GassServer.READ_ENABLE |
              GassServer.CLIENT_SHUTDOWN_ENABLE;
server.setOptions(options);

/**
 *Start the server on the specified host
 */

String resourceManagerContact = new String("hot.mcs.anl.gov");
server.start(resourceManagerContact);

/**
 * Get the URL for the remote server
 */

String url = server.getURL();

/**
 * Later - Shut down the remote server.
 */

server.shutdown();
```

Remote file I/O with GASS

Once you start a GASS server remotely, you can get input and output streams to read and write data for any remote file you have access to.

```
/**
 * Get the host and port information of the remote
 * GASS server, created in the previous section and
 * stored in an object called server.
 */

URL remoteGassUrl = new URL(server.getURL());
String host = remoteGassUrl.getHost();
int port = remoteGassUrl.getPort();

/**
 * Create an input stream to read data from a remote
 * file.

 * filepath - string containing absolute path of the
 * remote file.
 * For example,
 * /home/albert/foo.txt, for a Unix host
 * /c:/temp/myfolder/document.txt, for a Windows host
 */

GassInputStream in = new GassInputStream(host,
                                port, filepath);

/**
 * Read 10 bytes starting at offset 0 from this stream
 */

byte[] buf = new byte[10];
in.read(buf, 0, 10);

/**
 * GassInputStream supports some other functions,
 * like available() and getSize().   Please refer to
 * the Javadocs documentation for the usage
 * information of these methods
 */

/**
 * Close the stream
 */

in.close();

/**
 * Create an output stream to write data to a remote file.

 * length - this parameter specifies the total size of
 *        the data you want to write. If unknown, use -1.
 */

boolean append = true;
GassOutputStream out = new GassOutputStream(host, port,
                                  filepath, length, append);

/**
 * Write 10 bytes starting at position 0 to this stream
 */

out.write(buf, 0, 10);
```

```
/**
 * Close the stream
 */

out.close();
```

5.3.6 Limitations of the Java CoG Kit

Currently, Java CoG Kit does not provide support for GASS cache. This means that the experimental Job Execution Service (Section 8.2) provided by the Java CoG Kit does not cache executable and data files staged from clients. Also, the Java CoG Kit does not provide any replacement for the `globus-gass-cache` command line utility available in the Globus Toolkit.

6 Job Submission

This chapter provides information about job submission using the Java CoG Kit. Job submission in the Java CoG Kit is done using the Globus Resource Allocation Manager (GRAM).

6.1 Introduction

The Globus Resource Allocation Manager processes the requests for resources for remote application execution, allocates the required resources, and manages the active jobs. The Java CoG Kit provides a GRAM API for submitting and canceling a job request, as well as checking the status of a submitted job. The job specifications are written by the user in the Resource Specification Language (RSL) and are processed by GRAM as part of the job request. The GRAM service is mainly provided by a combination of two programs: the gatekeeper, and the job manager. When a job is submitted, the request is sent to the gatekeeper of the remote computer. The gatekeeper handles the request and creates a job manager for the job. The job manager starts and monitors the remote program, communicating state changes back to the user on the local machine. When the remote application terminates, successfully or by failing, the job manager terminates as well. GRAM is responsible for the following:

- Parsing and processing the Resource Specification Language (RSL) specifications that specify job requests.
- Job process creation and job control.
- Enabling remote monitoring and managing of jobs already created.

6.1.1 Gatekeeper

The gatekeeper is a remote service that authenticates and authorizes the execution of a service. It receives requests from clients, and performs mutual authentication with the client. After authenticating and authorizing it starts a job manager running under the credentials of the authenticated user. A gridmap file is used by the gatekeeper to map Globus credentials to local users. Figure 6.1 shows a schematic representation of this process.

The Java CoG Kit provides a personal gatekeeper that can be used as a lightweight alternative to the Globus gatekeeper. Details about the differences between the personal gatekeeper and the Globus gatekeeper can be found in Section 6.4.1

6.1.2 Job Manager

A job manager is spawned by the gatekeeper upon receiving each request. The job manager processes job specifications sent by the clients, most of which result in a job submission to a local scheduler. It also provides a mechanism through which the client can check the status of a job or cancel it. More information about the job manager can be found at [14].

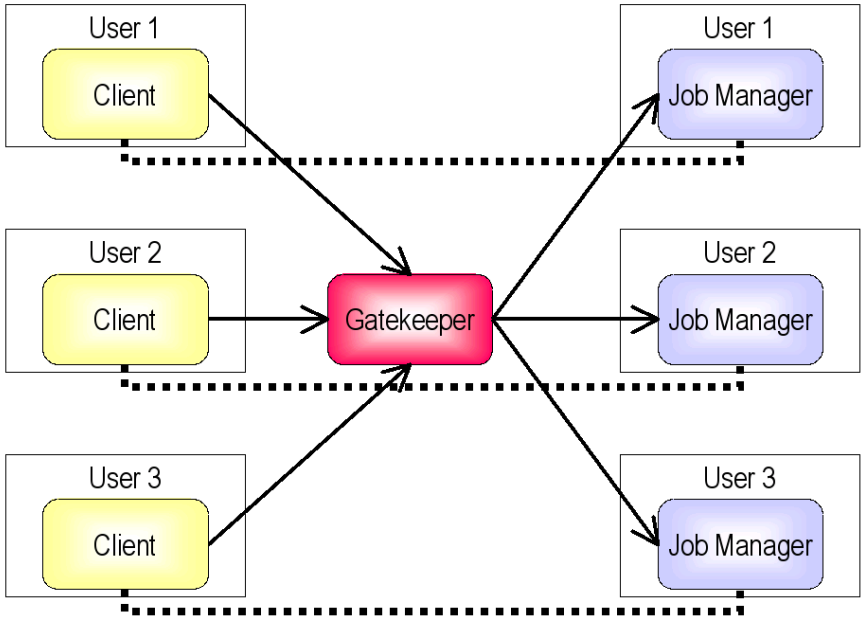

Figure 6.1: Gatekeeper Architecture

6.1.3 Batch and Interactive Jobs

Job execution can be done in two major ways: batch and interactive. Interactive jobs provide immediate feedback to the user. With interactive jobs the input can be redirected from a file, whether local or remote. The output and error streams of remote jobs are redirected to remote files, which can be monitored from the local machine. In contrast, batch jobs have their output/error streams stored into remote files, which can be retrieved after the job completes. Batch jobs are suitable when immediate feedback from the job is not needed, when multiple jobs are launched in parallel, or when the execution time is expected to be very large.

6.1.4 File Staging

GRAM also provides the ability to stage in data or executables, using a facility called Global Access to Secondary Storage (GASS). File staging allows you to automatically transfer any files required by your job, from the client machine, to the server machine. It is also possible to transfer the output files back to the client machine, after the job ends. Details about GASS can be found at [10]

6.2 Globus Resource Specification Language (RSL)

RSL is a common interchange language used to describe resources, irrespective of the scheduler or batch system used. RSL provides skeletal syntax to describe resources and various resource management components resulting in <attribute, value> pairs. Each attribute in the resource description serves as a parameter to control the behavior of one or more components in the resource management system.

6.2.1 RSL Syntax

The core syntax of the RSL is the relation of the form <attribute, value> pair. e.g "executable=a.out". More complicated resource descriptions can be build from the basic relations using compound requests and value sequences. Compound request can be formed using conjunction, disjunction or multi-request. Value sequences are used to express ordered sets of values. The value sequence syntax is used primarily for defining variables and for providing the argument list for a program. The & operator can be used to denote a conjunct request.

RSLAttributes

Following is a list of commonly used attribute names used in conjunction with GRAM:

executable : - describes the application to be executed

directory : - represents the remote working directory used for the execution of the job

arguments : - sets the command line arguments that will be passed to the executable

stdin : - allows input redirection for the job from a file.

stdout : - allows output redirection for the job

stderr : - specifies the redirection of the error stream

For a complete set of GRAM attributes please consult the following link:

GRAM - RSL parameters: : http://www.globus.org/gram/gram_rsl_parameters.html

Examples

- Typical GRAM resource descriptions contain at least a few relations in a conjunction:

```
(* this is a comment *)
& (executable = a.out (* <-- that is an unquoted literal *))
  (directory  = /home/albert )
  (arguments  = arg1 "arg 2")
  (count = 1)
```

- Substitutions can be used to make sure the same substring is used multiple times in a resource description:

```
& (rsl_substitution  = (TOPDIR  "/home/albert")
                (DATADIR $(TOPDIR)"/data")
                (EXECDIR $(TOPDIR)/bin) )
  (executable = $(EXECDIR)/a.out
  (directory  = $(TOPDIR) )
  (arguments  = $(DATADIR)/file1
  (environment = (DATADIR $(DATADIR)))
  (count = 1)
```

This is equivalent to the following RSL string:

```
& (rsl_substitution  = (TOPDIR  "/home/albert")
                (DATADIR "/home/albert/data")
                (EXECDIR "/home/albert/bin") )
  (executable = "/home/albert/bin/a.out" )
  (directory  = "/home/albert" )
```

```
(arguments  = "/home/albert/data/file1")
(environment = (DATADIR "/home/albert/data"))
(count = 1)
```

6.2.2 RSL in the Java CoG Kit

The Java CoG Kit RSL Parser used by the Java CoG Kit job manager does not support the full functionality of the C Globus RSL parser. Details about the differences can be found in Section 6.4.

6.3 Job Submission

Some of the tools mentioned in this section require that you have the environment variable `COG_INSTALL_PATH` set. For details on configuring the Java CoG Kit please refer to Chapter 3.6.

The executables described in this section can all be found in the `<cog-install-path>/bin` directory.

6.3.1 GUI

Form

A simple graphical interface is available by executing:

```
> cog-form
```

The program allows you to specify job parameters in a convenient form. A sample screen-shot is shown in Figure 6.2.

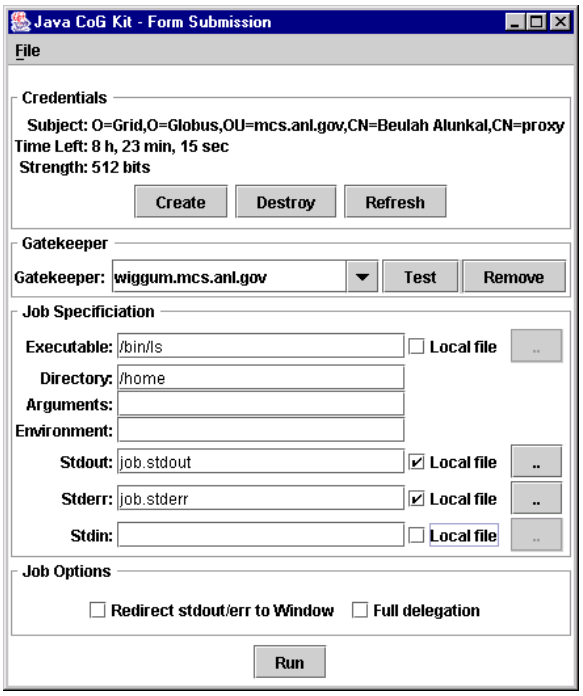

Figure 6.2: The CoG Form

Drag and Drop Desktop

A more experimental and hopefully more intuitive interface for submitting jobs can be started by executing:

```
> cog-desktop
```

With Drag and Drop Desktop multiple jobs and servers can be configured graphically. A job submission is a simple matter of dragging the icon of a job over the icon of a configured server. A Drag and Drop Desktop sample screen-shot can be seen in Figure 6.3.

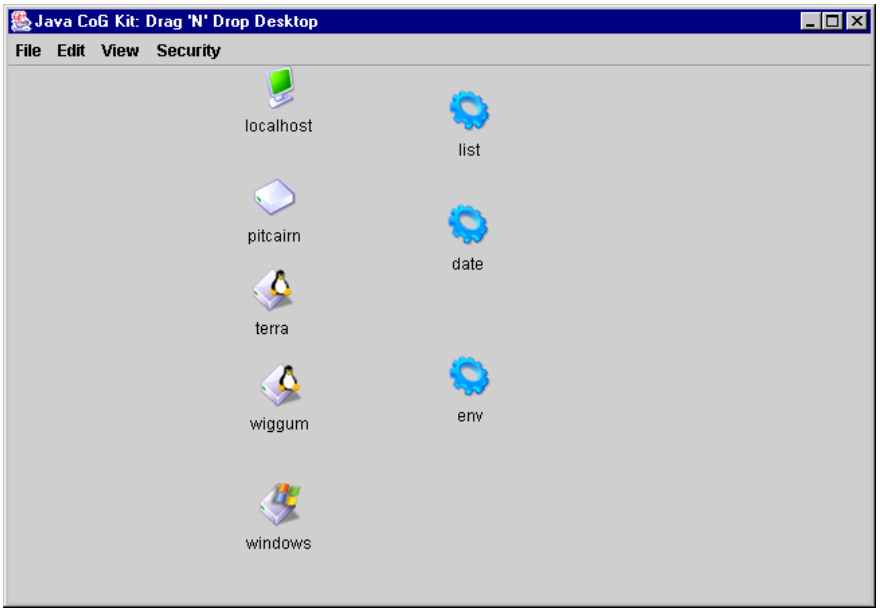

Figure 6.3: The Drag and Drop Desktop

6.3.2 Unix Shell Scripts

You can use `globusrun` to execute remote jobs from the command line. The format for running `globusrun` is:

```
> globusrun [options] [RSL string]
```

For a complete list of options accepted by `globusrun` please run the following command:

```
> globusrun -help
```

A simple example which lists the current directory on the remote machine and prints the result on the client machine is provided below:

```
> globusrun -r hot.mcs.anl.gov -o "&(executable=/bin/ls)"
```

The `-r` parameter allows you to specify the remote machine to which the job is being submitted. The `-o` parameter instructs both the client and the server to treat this job as an interactive job, redirecting the input and the output from and to the client machine.

6.3.3 Windows Batch Files

In Windows, the `globusrun.bat` file can be used to execute a remote job from the command line. The syntax of the command line is identical to the one for the Unix shell script. Please refer to the previous subsection for details.

6.3.4 Java CoG Kit Shell

The Java CoG Kit Shell is a convenience application that allows you to use several Java CoK Kit features from a command-line like interface. To start the shell execute the following from the <`cog-install-path`>`/bin` directory:

```
> cog-shell
```

From inside the console you can use the `globusrun` command. The syntax and options for the `globusrun` command inside the console are identical to those of the Unix `globusrun` shell command, or the Windows `globusrun.bat` batch.

6.3.5 Job Submission API

The Java CoG Kit provides an extensive API for handling the execution of jobs, and other tasks associated with job execution.

Remote Executables

A remote executable can be executed by simply passing its name through the RSL description:

```
& (executable = /bin/ls)
```

Submitting the Job

After the RSL description was built, it must be submitted to the server. First you should ensure that the gatekeeper is alive on the remote machine:

```
Gram.ping("hot.mcs.anl.gov");
```

Next, a `GramJob` object is instantiated, passing the RSL string to the constructor.

```
GramJob job = new GramJob(RSLString);
```

Feedback from the remote server is provided in order to interact with the job. A listener can be used to receive notifications about job status from the server:

```
class GramJobListenerImpl implements GramJobListener {
        public void statusChanged(GramJob job) {
                String status = job.getStatusAsString();
        }
job.addListener(new GramJobListenerImpl());
```

The job is now ready for submission. The actual submission is done through the `request` method, which takes two arguments:

- The first argument specifies the remote server
- The second argument indicates whether the job is submitted in batch or interactive mode. A value of `true` denotes batch mode.

```
job.request("hot.mcs.anl.gov", false);
```

Local Executables and File Staging

By default the job manager will look for the executable and input/output files on the remote machine on which the job is scheduled for execution. In case the executable file resides on the local machine, or if the job requires a local file as input, file staging needs to be used. In order to use file staging, a GASS server needs to be started on the local machine. This can be done by instantiating a new `GassServer` object. The first argument passed to the constructor enables or disables the starting of the GASS server in secure mode. In the example below, the GASS server is started in secure mode. The second argument indicates the port on which the GASS server will listen to incoming connections. If the second argument is set to zero, a port will be chosen automatically. The `getURL()` retrieves the URL associated with the GASS server for further reference.

```
GassServer gass = new GassServer(true, 0);
String gassUrl = gass.getURL();
```

The resulting GASS URL must be passed to the GRAM server through the RSL description for the job.

Suppose your gassUrl is `https://140.221.10.38:4678` and you want to run an executable called `c:\a.exe`. The resulting RSL description would contain the following:

```
& (rsl_substitution = (GLOBUSRUN_GASS_URL https://140.221.10.38:4678))
  (executable = $(GLOBUS_GASS_URL)/c:/a.exe)
```

Retrieving Output and Error Messages for Jobs

For batch jobs, the output and error streams are redirected to remote files, which are not retrieved after the job terminates. To avoid this, output and error streams can be redirected to the client machine using GASS.

To redirect the output to the client machine, you need to pass the GASS URL to the server through the `stdout` and `stderr` parameters in the job RSL. This will stream the job output and error streams to the local GASS server.

```
GassServer gass = new GassServer(true, 0);
rsl = new RslAttributes();

...

rsl.add("stdout", gass.getURL() + "/dev/stdout");
rsl.add("stderr", gass.getURL() + "/dev/stderr");
```

Register the JobOutputListener class with the Gass server.

```
JobOutputListenerImpl outListener =  new JobOutputListenerImpl();
JobOutputStream outStream = new JobOutputStream(outListener);
gass.registerJobOutputStream("out", outStream)
gass.registerJobOutputStream("err", outStream)

class JobOutputListenerImpl implements JobOutputListener {
        public void outputClosed() {
                //Job has finished; no more output is available
        }
  };
```

Sample code

Sample program testing various features is available at:

jglobus : `src/org/globus/gram/Gram15Test.java`

It can be run from jglobus directory using the following ant command

```
> ant -buildfile progs.xml GramTest3 <machine name>
```

6.4 Differences from the C Globus Toolkit

6.4.1 Gatekeeper

Due to the lack of operating-system specific programming interfaces of the Java programming language, the personal gatekeeper does not allow user remapping. Hence, the job managers spawned by the personal gatekeeper can only run with the same priviledges as the gatekeeper itself.

6.4.2 RSL Parser

The following features avaiable in the Globus C RSL parser are not supported by the Java CoG Kit RSL parser:

1. User-specified delimiter for quoted literals.
2. RSL strings that only contain relations outside of specifications.

7 Accessing the Grid Information Service

This chapter gives a brief overview of the Grid Information Service architecture and explains the different ways in which a user can access the Grid Information Servers using the Java CoG Kit.

7.1 Introduction

Grid technologies enable large-scale sharing of resources within groups of individuals and organizations. In these settings the user might be interested in discovering and monitoring the resources in a secure and efficient way. The Globus Toolkit supports a Monitoring and Discovery Service (MDS)[15] to provide information about Grid resources. In short, MDS provides directory services for resources in the Grid. A directory service provides information about different entities in the environment (such as resources and services) to applications and their users.

Extensive documentation for MDS is available at

Homepage : `http://www.globus.org/mds`

Manual : `http://www.globus.org/mds/mdsusersguide.pdf`

More technical details are available at [16]

7.2 Architecture

The structure of MDS is hierarchical [17]. It consists of Grid Resource Information Service (GRIS), Grid Index Information Service (GIIS) and Information Providers (IPs) as shown in Figure 7.1.

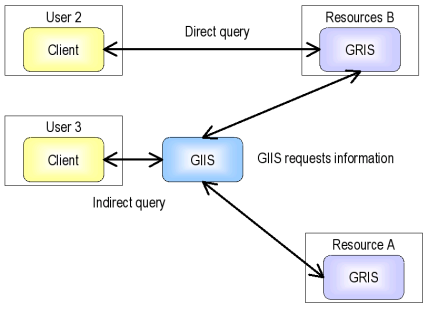

Figure 7.1: Architecture of MDS.

7.2.1 GRIS

GRIS is information service that runs on a single resource and can answer queries from a user about that particular resource by directing these queries to an information provider deployed on that resource.

7.2.2 IPs

An Information Provider (IP) is a service that generates information about a specific aspect of a resource. The query from GRIS to a resource could be requesting any or all of the following types of data:

- Platform type and architecture.
- Operating system: host OS and version.
- CPU information: type, number of CPUs, version, speed, cache.
- Physical and virtual memory: size and free space.
- Network interface information: machine names and IP addresses.
- File system summary: size, free space.

The following link gives a set of core Information Providers available for MDS.

Information Providers : `http://www.globus.org/mds/DefaultGRISProviders.html`

7.2.3 GIIS

A GIIS is an aggregate directory service that can supply a collection of information gathered from multiple GRIS and GIIS resources available at a site. It supports queries against information spread across multiple GRIS resources.

7.2.4 Working

Every resource running MDS has a GRIS. A GRIS can respond to queries from other systems on the Grid asking for information about a local machine or other specific resource. It can be configured to register itself with aggregate directory services such as GIIS (so that those services can pass on information about the machine to others). GRIS authenticates and parses each incoming information request and then dispatches those requests to one or more local Information Providers, depending on the type of information named in the request. Results are then sent back to the client.

In order to get collective information about two or more resources present in a single site, the queries can be sent directly to GIIS. In that case the GIIS directs the query to GRIS.

7.3 Security with MDS

MDS uses the Grid Security Infrastructure (GSI) which enables the use of certificates to provide authentication and authorization. MDS provides both authenticated as well as anonymous accesses by the users. For authenticated access to MDS, the user requires a user certificate and certain other credentials as described in Section 4.2.1.

7.3.1 Site Policies

The Site Policies specify the restrictions on registration of resources with GIIS by the system administrator. An open policy for a GIIS allows all the GRIS or GIIS resources to be registered with it. Whereas in a closed system only specified resources can register with a GIIS. The default is for the GIIS to accept registrations

only from itself. By default the GIIS service runs on port 2135. Please contact your system administrator for your local site policies.

7.4 Accessing Grid Information Services

This section explains the different methods provided by the Java CoG Kit to access the Grid Information Services.

7.4.1 Using Graphical User Interface (GUI)

The LDAP Browser/Editor provides a user-friendly Windows Explorer-like interface to LDAP directories with tightly integrated browsing and editing capabilities. It is entirely written in Java with the help JNDI class libraries. It can connect to LDAP v2 and v3 servers. Figure 7.2 shows the user interface of the LDAP browser/editor.

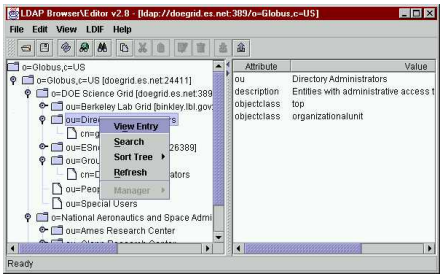

Figure 7.2: LDAP Browser

Homepage : `http://www.mcs.anl.gov/~gawor/ldap/`

Out of historical reasons the browser is not distributed with the Java CoG Kit. This may change in the future.

Using Web Browser

User-friendly Web browser access to Information Services can also be provided through a set of PHP scripts on a PHP-enabled Web server. These PHP scripts can be added to any Web page and perform MDS queries to gather basic information. The scripts can be easily adapted to show the summary data needed by a project. For information on PHP, please refer to the following link.

PHP : `http://www.php.net/`

Example Web Interface : The Globus project maintains an MDS index node (giis) available for anyone to query. The web interface for this node is available at the following link.

Globus-giis : `http://giis.globus.org/ldapbrowser/login.php`

7.4.2 Unix Shell scripts

The Java CoG Kit provides a Unix shell script `grid-info-search` in the `<cog-install-path>/bin` directory. The tool has the following syntax.

```
> grid-info-search [options]  search_filter  [attributes]
```

The usage messages for this command are available the Appendix A of this manual. The following examples describe some of the ways of using the `grid-info-search` command:

Query all objects on GRIS :

This example shows how to display all of the data objects and resources on a single machine set up as a GRIS. Assume the machine `hot.mcs.anl.gov` has a GRIS service running at port `2135` . The command will be given as follows:

```
> grid-info-search -x -h hot.mcs.anl.gov -p 2135
-b "Mds-Vo-name=local, o=Grid" "(objectclass=*)"
```

The option `-x` is used to denote anonymous access and `-b` denotes the branch point. A branch point is the location in the directory from which to start the search. The default branch point for GRIS service in MDS is `Mds-Vo-name=local, o=grid`. The final argument in this example is the search filter. It specifies the category of object class you wish to search. The search filter `(objectclass=*)` here indicates that the information regarding all the object classes needs to be displayed. Please refer to Section 7.5 for the syntax and attributes of the objectclass. A part of the output for the above query would look as follows:

```
dn: Mds-Host-hn=hot.mcs.anl.gov, Mds-Vo-name=local, o=Grid
Mds-Cpu-speedMHz: 866
Mds-Memory-Ram-Total-freeMB: 304
Mds-Fs-freeMB: 10
Mds-Fs-freeMB: 21
Mds-Fs-freeMB: 270
Mds-Fs-freeMB: 341
Mds-Fs-freeMB: 4428
Mds-Fs-freeMB: 47
Mds-Fs-freeMB: 73
Mds-Cpu-Free-5minX100: 134
Mds-Net-Total-count: 2
Mds-validfrom: 20030303165825Z
 Mds-Cpu-Total-count: 2
Mds-Memory-Vm-sizeMB: 243
Mds-Cpu-vendor: GenuineIntel
Mds-Net-name: eth0
Mds-Net-name: lo
Mds-validto: 20030303165825Z
```

Query file system space on a GIIS :

This example shows how to query for the amount of free file system space on all machines on a GIIS running for a site. The command is as follows:

```
> grid-info-search -h giis.mcs.anl.gov -p 2135 -b
 "Mds-Vo-name=site,o=Grid" "(objectclass=*)" Mds-Fs-freeMB
```

Here it is assumed that GIIS is running on machine `giis.mcs.anl.gov` at port `2135`. The branch point option `-b` has the value `Mds-Vo-name=site, o= grid`. This is the default branch point for GIIS server. The attribute `Mds-Fs-freeMB` specifies that the information regarding the amount of free

file system space, on all the machines registered on the GIIS, needs to be displayed. Assume that cold.mcs.anl.gov and hot.mcs.anl.gov are registered on GIIS. Part of the output would look as follows:

```
dn: Mds-Host-hn=hot.mcs.anl.gov, Mds-Vo-name=site,o=Grid
Mds-Fs-freeMB: 10
Mds-Fs-freeMB: 21
Mds-Fs-freeMB: 270
Mds-Fs-freeMB: 341
Mds-Fs-freeMB: 4388
Mds-Fs-freeMB: 47
Mds-Fs-freeMB: 73

dn: Mds-Host-hn=cold.mcs.anl.gov, Mds-Vo-name=site,o=Grid
Mds-Fs-freeMB: 10
Mds-Fs-freeMB: 21
Mds-Fs-freeMB: 270
Mds-Fs-freeMB: 341
Mds-Fs-freeMB: 4388
Mds-Fs-freeMB: 47
Mds-Fs-freeMB: 73
```

Query CPU data on a single machine on a GIIS : This example shows how to query for CPU model and speed on a single machine on a GIIS. The command is as follows:

```
> grid-info-search -x -h giis.mcs.anl.gov -p 2135 -b
"Mds-Vo-name=site, o=Grid"
"(&(objectclass=MdsCpu)(Mds-Host-hn=cold.mcs.anl.gov))"
Mds-Cpu-model Mds-Cpu-speedMHz
```

Here we are querying a GIIS server, but we specify the name of a single machine `cold.mcs.anl.gov` in which we are interested. So it retrieves the CPU model and speed of that singe machine only. The output for the above query is given below.

```
dn: Mds-Host-hn=cold.mcs.anl.gov, Mds-Vo-name=site, o=Grid
Mds-Cpu-model: Pentium III (Coppermine)
Mds-Cpu-speedMHz: 866
```

7.4.3 Windows batch files

The grid-info-search batch file available for windows machines available in the `<cog-install-path>/bin` directory performs the same way as discussed in Section 7.4.2

7.4.4 Using the API to access MDS

Netscape Directory SDK and JNDI (with LDAP the provider) are libraries that can be used to retrieve resource information from an MDS server. The Netscape API is LDAP-specific. It is used for low-level access to LDAP directories. JNDI is a generic API for retrieving directory information.

In addition to these libraries, there is an MDS library distributed with the Java CoG Kit. Although it is deprecated, we still provide it to maintain backward compatibility. It is a simple layer built on top of JNDI with LDAP-specific calls. It

was originally written as a work around for some bugs found in early versions of JNDI. However, JNDI is much more stable now. It provides a powerful and flexible interface to directory based services and is more appropriate for accessing MDS.

The MDS server allows both anonymous and authenticated access to the resource information. Anonymous access does not require the user to have any specific credentials. Details of how to connect as anonymous or authenticated user using both the Netscape and JNDI libraries are explained in the following subsections.

Anonymous Access Using JNDI/LDAP library:

Here we describe the how to access MDS anonymously using JNDI. The tutorial for using JNDI is available at the following link:

JNDI : http://java.sun.com/products/jndi/tutorial/TOC.html

Setting up a connection with MDS using JNDI includes the following steps:

```
/*  Step 1: Provide the host and port information of
 *  the MDS server that is to be queried.
 */

Hashtable env = new Hashtable();
env.put(Context.PROVIDER_URL, "ldap://"+ host + ":" + port);

/* Step 2: Specify the anonymous access
 */

env.put(Context.SECURITY_AUTHENTICATION, "simple");

/* Step 3: Create the Initial Dir Context
 */

DirContext ctx = null;
ctx = new InitialDirContext(env);

/* Step 4: Search for required information
 */

String baseDN  = "mds-vo-name=local, o=grid";
String filter  = "(objectclass=*)";
NamingEnumeration results = ctx.search(baseDN, filter, null);

/* Step 5: Display the results
 */

SearchResult si;
Attributes attrs;
while (results.hasMoreElements()) {
 si = (SearchResult)results.next();
 attrs = si.getAttributes();
 System.out.println(si.getName() + ":");
 System.out.println(attrs);
 System.out.println();
}
```

Anonymous Access Using the Netscape Directory SDK:

Anonymous access to MDS is described in this section using the Netscape Library. Please make sure to include the Netscape library jar file in your classpath before

compiling your programs. You can get the jar file by following the instructions provided in the given link:

Netscape : http://www.mozilla.org/directory/javasdk.html

A patched version of the Netscape library `ldapjdk-patched.jar` is distributed with the Java CoG Kit in the `src/org/globus/mds/gsi/netscape/` directory. You can use this jar file. The patched library has a bug fix for certain security related issues.

Setting up an anonymous connection using Netscape Directory SDK includes the following steps.

```
/* Step 1: Create an LDAPConnection
 */

String binddn  = null;

LDAPConnection ld = null;
ld = new LDAPConnection();

/* Step 2: Connect to host and port
 */

ld.connect( host, port );

/* Step 3: Retrieve the results
 */

String baseDN  = "mds-vo-name=local, o=grid";
String filter  = "(objectclass=*)";
LDAPSearchResults myResults = null;
myResults = ld.search( baseDN, LDAPv2.SCOPE_ONE,
                         filter, null, false );

/* Step 4: Display the results
 */

while ( myResults.hasMoreElements() ) {
 LDAPEntry myEntry = myResults.next();
 String nextDN = myEntry.getDN();
 System.out.println( nextDN + ":");
 LDAPAttributeSet entryAttrs = myEntry.getAttributeSet();
 System.out.println(entryAttrs);
 System.out.println();
}
```

Authenticated Access to MDS

The org.globus.mds.gsi library provides bindings for both Netscape Directory SDK and JNDI (with LDAP provider) for establishing secure connection with GSI-enabled ldap servers such as an MDS-2 server. The bindings are based on the SASL protocol, defined in the RFC document available at the following link.

RFC : http://www.ietf.org/rfc/rfc2222.txt

The library is used in the same manner as any other SASL mechanism. The only differences are the properties that can be passed to the underlying SASL mechanism. The properties that needs to be set in order to use GSI while using the Netscape Library or the JNDI library are given below:

1. javax.security.sasl.client.pkgs:

   This property has to be set to “org.globus.mds.gsi.netscape” while using Netscape Directory SDK and “org.globus.mds.gsi.jndi” while using the JNDI/LDAP provider. It basically specifies the package that provides implementation for the SASL mechanism.

2. javax.security.sasl.qop:

   It specifies what Quality-Of-Protection (QOP) to use. It is a list of QOP values put in order of preference. Allowed QOP values are :

auth : - authentication only

auth-int : - authentication with integrity protection (GSI without encryption)

auth-conf : - authentication with integrity and privacy protections (GSI with encryption)

If the property is not specified, it defaults to ”auth”.

3. javax.naming.Context.SECURITY_CREDENTIALS:

   It specifies the credentials to use for SASL authentication. If the property is not set, the default credentials will be used.

4. javax.security.sasl.strength:

   It specifies the strength of encryption. But it is currently not used by the library.

Authenticated Access Using JNDI/LDAP library:

In order to establish authenticated access to MDS using JNDI, you need version 1.2.3 or above of the JNDI/LDAP library. For setting up a secure connection with MDS replace the Step 2 of Anonymous Access Using JNDI/LDAP library Section 7.4.4 with the following steps:

```
/* This property specifies where the implementation of
 * the GSI SASL mechanism for JNDI can be found.
 */

env.put("javax.security.sasl.client.pkgs",
                "org.globus.mds.gsi.jndi");

/* This property specifies the quality of protection
 * value.
 */

env.put("javax.security.sasl.qop", "auth" );

/* Specify the particular SASL mechanism to use.
 */

env.put(Context.SECURITY_AUTHENTICATION, "GSIMechanism.NAME");
```

Authenticated Access Using Netscape Directory SDK:

In order to establish a secure connection using Netscape library, you need to use Version 4.1 or above of the Netscape Directory SDK.

To provide authenticated access instead of anonymous access include these steps after Step 2 in the Anonymous Access Using the Netscape Directory SDK Section 7.4.4

```
Hashtable props = new Hashtable();

/* This property specifies where the implementation of
 * the GSI SASL mechanism for Netscape Directory SDK
 * can be found.
 */

props.put ( "javax.security.sasl.client.pkgs",
                    "org.globus.mds.gsi.netscape" );

/* This property specifies the quality of protection
 * value
 */

props.put("javax.security.sasl.qop", "auth" );

/* Authenticate to the server over SASL.
 */


ld.authenticate( null, new String [] {"GSIMechanism.NAME"},
                                          props, null );
```

Example Location

The same example program for the Netscape using GSI security mechanism is available at the following location:

jglobus : `org/globus/mds/gsi/NetscapeTest.java`

For JNDI example program using GSI, please refer to the following program location:

jglobus : `org/globus/mds/gsi/JndiTest.java`

Adding/Updating Entries using API in MDS

MDS is a READ ONLY directory service. Nevertheless you can add or update entries in MDS using the API, if the MDS server supports a backend read-write database. In that case, you populate the backend database yourself. Normally the backend is automatically populated by information providers. For further information please refer to the following link:

FAQ-UpdateMds : `http://www.globus.org/mds/FAQ.html#adddatatomds`

7.5 Schema

The information model used in MDS is based on entries arranged in an hierarchical tree-like structure. The tree is called the Directory Information Tree and the contents include object classes and entries. Object classes describe what information can be stored in the directory. The values of the object class determine the schema

rules the entries must obey. Few of the descriptions of the schema object classes and their attribute types are shown below:

```
Object class Mds
Attribute type Mds-validfrom
Attribute type Mds-validto
Attribute type Mds-keepto?

Object class MdsHost
Attribute type Mds-Host-hn

Object class MdsOs
Attribute type Mds-Os-name+
Attribute type Mds-Os-release+
Attribute type Mds-Os-version+

Object class MdsCpu
Attribute type Mds-Cpu-vendor+
Attribute type Mds-Cpu-model+
Attribute type Mds-Cpu-version+
Attribute type Mds-Cpu-features*
Attribute type Mds-Cpu-speedMHz*
```

For a detailed description of object classes and their attributes refer to the following webpage:

Schemas : `http://www.globus.org/mds/Schema.html`

For the syntax of the schemas, refer to RFC 2252 available at the following location:

Syntax : `http://www.ietf.org/rfc/rfc2252.txt`

7.6 Performance issues with MDS

The performace of a query depends upon the Information Providers used and the amount of time the data is live and cached. When a query to a GRIS arrives, it will be answered very quickly if the data requested is live and cached. If the data requested has been flushed from the cache because it has expired, the GRIS server will invoke the information providers to fetch the information. The time taken to deliver depends on the time taken by these providers.

The performance of a query to a GIIS is dependent upon the performance of the GRIS's that it accesses as well as the amount of time the data is live and cached. When a query to a GIIS arrives, it will be answered very quickly if the data is present in the cache. Otherwise the GIIS might query a GRIS that supplies the information.

In short, there is no appropriate formula for predicting the performance for a query to MDS. As the GIIS hierarchy becomes more complex, the performance becomes more unpredictable.

The performance of IPs have a great impact on the performance of a query in general. It is possible to write a (server-side) MDS information provider executable in Java. Nevertheless you might need to consider the JVM startup cost and other performance issues.

7.6.1 Programming Issues

[NEW, 15 March, 2002:

Retrieving information from MDS should be performed with thought and care. You should be connecting to the MDS server only as long as the connection is required in order to avoid blocking the limited number of ports to an MDS server. Hence it is better to disconnect from the server immediately. As a connection takes usually some time it is sometimes better to perform a number of queries. However you should avoid analysing the result between subsequent queries. Instead you should analyse the queries once all queries have been performed or start a parallel thread.

Wrong : This method blocks the port unnecessarily.

1. Connect to the server
2. Query the server
3. *Analyse* and Display the results
4. Disconnect from the server

Right : This is the prefer ed method.

1. Connect to the server
2. Query the server
3. Disconnect from the server
4. *Analyse* and Display the results

For iterative procedures we recommend the same:

Wrong :

1. Connect to the server
2. Query the server
3. *Analyse* and Display the results
4. goto 2 until all queries done
5. Disconnect from the server

Right :

1. Connect to the server
2. Query the server
3. goto 2 until all queries done
4. Disconnect from the server
5. *Analyse* and Display the results

Additionally, a user need to think about the correlation between query frequency and information update frequency of a value in the MDS. For example, if a user requests every second information that is only updated every thirty seconds this will lead to a waste of resources. We encourage Grid programmers to avoid such situations by having a clear understanding on how the information is updated. You can find out more about this from the MDS web pages.

7.7 Implementation Details of MDS 2.2 version

MDS 2.2 uses OpenLDAP Version 2.0.22, which implements LDAP Version 3. The security in OpenLDAP is provided by the Simple Authentication and Security Layer (SASL), which also uses GSS-API. SASL is a method for adding authentication support to connection-based protocols. MDS 2.2 uses Cyrus SASL Version 1.5.27. SASL is a convenient generic interface for secure application development. By itself, SASL does not provide any security. It relies on underlying technologies to provide the actual identity authentication and message protection services desired by applications communicating over a network. Applications may install and request the use of particular mechanisms or use a default mechanism provided by the SASL implementation.

MDS 2.2 also uses OpenSSL Version 0.9.6b. OpenSSL provides the Secure Sockets Layer (SSL) implementation used by the GSI. OpenSSL is an open-source implementation of SSL used to build the GSS-API.

7.8 Differences between Java and Globus tool

The command-line arguments for the `grid-info-search` in Java CoG are slightly different from those of the C version.

For example, for the `grid-info-search` command we have not enabled `-config file` option that specifies a different configuration file to obtain MDS defaults and `-nowrap` option that passes the output through a line-unwrapping filter first.

In Java CoG Kit implementation, the search filter needs to be specified in order to get the results. Whereas this is optional in C version. For details please check `grid-info-search -help` in both Java and C versions.

8 Server-side Java CoG Kit

This chapter gives an overview of how to start up Job Execution and File Transfer Services present in Java CoG Kit.

8.1 Introduction

The Java CoG Kit provides client side as well as partial server-side functionality for enabling operations on Grid. While the other chapters of this manual focus on the client side functionality, this chapter focuses on the server side functionality. The Java CoG Kit provides experimental implementations of a Job Execution Service and a File Transfer Service. Job Execution service includes a Personal Gatekeeper and a Job Manager while the File Transfer service includes the GASS mechanism.

The Java CoG Kit does not include GridFTP server for file transfers, MDS server for storing and retrieving resource information and a full fledged Job Execution Service for executing jobs securely on remote machines. These services are provided by C Globus Toolkit. Detailed information of these C-based services can be found at the following links:

Globus-GridFTP : `http://www.globus.org/datagrid/gridftp.html`

Globus-GRAM : `http://www.globus.org/gram`

Globus-MDS : `http://www.globus.org/mds`

Details on how to start up these services for Globus Toolkit 2.2 are available at the following link:

Globus-install : `www.globus.org/gt2.2/admin/guide-startup.html`

8.2 Job Execution Service

Java CoG Kit contains an experimental and elementary Job Execution Service. The implementation includes a Personal Gatekeeper and a Job Manager.

A client submits a job request to the Personal Gatekeeper. The Personal Gatekeeper performs authentication with the client and starts a Job Manager. The Job Manager receives the job requests, interprets them and executes the jobs either interactively using the *fork* jobmanager or through batch schedulers such as *PBS, LSF*.

Normally a Gatekeeper has to map the identity of the client to a local user and start the Job Manager as that local user. But since the Java Virtual Machine allows only limited interaction with the Operating System, this functionality cannot be implemented in Java. As such the Personal Gatekeeper cannot map between the root and user id like the C Globus Gatekeeper. Hence this service can be used for personal grids or adhoc-grids controlled by a singe user. All jobs submitted will be executed with that user account settings.

Details about the full fledged Globus C-based Gatekeeper are available in Section 6.1.

8.2.1 Configuration

The Personal Gatekeeper supports two configuration files. One of them is used for configuring specific jobmanagers (such as fork , pbs, etc) and the other is a gridmap file, used for authorizing different users to use the service. Both these files can be specified either through the program or by using the command line tools which are described in the next section. If the configuration file for job managers is not specified the gatekeeper starts the default fork job manager. A sample configuration file for specifying job managers is available at

jglobus : `org/globus/gatekeeper/services.conf`

A gridmap file consists of single line entries listing a certificate subject and a userid, like this:

```
"/O=Grid/O=Globus/OU=your.domain/CN=Your Name" userid
```

where subject name refers to the subject that appears on your certificate and a userid refers to your account login name on the server machine. When a client connects to the gatekeeper, the subject of the certificate will be searched in the gridmap file. If it is not found, the connection is rejected. If the subject name is found the connection is allowed. This file need not be specified if the Gatekeeper is used by a single user. For more information on gridmap file refer to Section 4.2.5.

8.2.2 Limitations

Currently the implementation does not support:

- Caching of files (using gass cache)
- Running services as authenticated user (You can be authenticated. But no remapping is done by the Gatekeeper, therefore all jobs are run as the same user)
- Poe or mpirun for fork job manager, or condor submissions. However they can be performed using full delegation.

The implementation of the gatekeeper has a synchronization problem, due to which the output might not get appropriately streamed or redirected to the client. We recommend using it in batch mode without IO redirection.

8.2.3 Starting the personal gatekeeper

The Gatekeeper can be invoked in any of the following ways:

Command Line : The Java CoG Kit provides a Unix shell script and a window batch file to start up the personal gatekeeper. To start the Gatekeeper, run the script or batch file `globus-personal-gatekeeper` available in the `<cog-install-path>/bin` directory as follows.

```
> globus-personal-gatekeeper
```

Using the API : The Personal Gatekeeper can be started from within a program using the API provided in jglobus. A sample code is shown below.

```
/* Step 1:  Initialize the variables
 */

import org.globus.gatekeeper.GateKeeperServer;
GateKeeperServer gk = null;
GSSCredential gssCred = null;
String logFile = null;
String gridMapFile = null;
Properties props = null;
int port = GateKeeperServer.PORT;

/* Step 2:  Obtain the credentials
 */

GlobusCredential credentials = null;
credentials = GlobusCredential.getDefaultCredential();
gssCred = new GlobusGSSCredentialImpl(credentials,
                  GSSCredential.ACCEPT_ONLY);

/* Step 3:  Load the gridmap file if available.
 */

GridMap gridMap = new GridMap();
gridMap.load(gridMapFile);

/* Step 4:  Start the server
 */

gk = new GateKeeperServer(gssCred, port);

/* Step 5:  Set the gridmap and log files if needed.
 */

gk.setGridMap(gridMap);

if (logFile != null) {
        gk.setLogFile(logFile);
           }

/* Step 6:  Register with the job manager services.
 * Read it from the configuration file if specified
 * otherwise enter the service name directly as shown.
 */

if (props != null) {
   gk.registerServices(props);
} else {
   gk.registerService("jobmanager",
   "org.globus.gatekeeper.jobmanager.ForkJobManagerService",
    null);
}
```

The above example program is available at the following location:

jglobus : `org/globus/gatekeeper/Gatekeeper.java`

8.2.4 Differences between Java and Globus Personal Gatekeeper

The Java CoG Kit Personal Gatekeeper is compatible in many aspects to the Globus Personal Gatekeeper. Also, a globusrun tool from Globus 2.2.4 version can submit a job to the Java Personal Gatekeeper and get back the result. The differences include limitations in our implementations. They are given in Section 8.2.2.

8.3 File Transfer Service

Globus Access to Secondary Storage(GASS) is a mechanism used to transfer data using the HTTP protocol. A GASS server uses secure HTTP for authentication and data transfer. A GASS server could be run as part of a job submission to transfer standard input and output files and prestage executables to remote servers as explained in Section 6.3. It can also be used to transfer data as explained in Section 5.3

The Java CoG Kit provides client and server GASS functionality. It provides a pure Java Globus GASS server for transferring files via HTTPS. The server is multi-threaded and accepts HTTPS connection from GASS clients to copy from, copy to, and append to files that are local to the server. It also provides a pure Java Globus GASS client for transferring files via HTTPS.

8.3.1 Limitations

The GASS servers does not support the cache management functionality.

8.3.2 Starting the Gass Server

The GASS server can be invoked in any of the following ways.

Command Line : To start the GASS server, run the script or batch file `globus-gass-server` available in the `<cog-install-path>/bin` directory as follows.

```
> globus-gass-server
```

If you wish to shut down the server using the command line tool, you need to specify the option `-c` or `-client-shutdown` while starting the server. In that case, server can be shut down using the `globus-gass-server-shutdown`. Let us assume that gass server is started on machine named `hot.mcs.anl.gov` at port number `4573`. It can be stopped using the following command:

```
> globus-gass-server-shutdown  "hot.mcs.anl.gov:4573"
```

Using the API : It can be started from within a program using the API. A sample code is shown below.

```
/* Step 1:  Initialize the variables
 */

int port = 0;
boolean secure=true;
int options =
    org.globus.io.gass.server.GassServer.READ_ENABLE |
    org.globus.io.gass.server.GassServer.WRITE_ENABLE |
    org.globus.io.gass.server.GassServer.STDOUT_ENABLE |
    org.globus.io.gass.server.GassServer.STDERR_ENABLE;

/* Step 2:  Start up the server in secure mode at the given port
 */

org.globus.io.gass.server.GassServer gassserver =
   new org.globus.io.gass.server.GassServer(secure, port);

/* Step 3:  Set the appropriate Options
```

```
 */

gassserver.setOptions(options);

/* Step 4:  Display the GASS url
 */

System.out.println(gassserver.getURL());
```

The example program is available at the following location:

jglobus : `org/globus/tools/GassServer.java`

In order to shut down the GASS server url using its url use the following code.

```
/* Step 1:  Create a GlobusURL
 */

String url=null;
GlobusURL gassURL = new GlobusURL(url);

/* Step 2:  Shut down the server
 */

org.globus.io.gass.server.GassServer.shutdown(null, gassURL);
```

The example program is available at the following location:

jglobus : `org/globus/tools/GassServerShutdown.java`

8.3.3 Differences between Java and Globus GASS service

The Java CoG Kit GASS implementation is compatible with Globus GASS. It allows, for example, a Java GASS client to connect and transfer a file from a Globus GASS server; or a Globus GASS client to connect and transfer a file from a Java GASS server. There are certain limitations in the implementation. They are given in Section 8.3.1.

9 Production Tests with the Java CoG Kit

[new March 28, 2003]

9.1 Introduction

Testing is a significant part of contemporary software development practices. With a proper design it can uncover a high percentage of problems before software is released. Tests can also be used with released software to reveal compatibility issues.

The Java CoG Kit contains two testing methodologies. First, it contains a number of unit tests that are run prior to a release to increase the code correctness. Second, it contains a number of production tests that are intended to check if elementary tasks such as job submission and filetransfer can be performed. In this section we concentrate on the later.

The Java CoG Kit production testing framework is designed to perform production tests in a flexible manner. It tests multiple Java Development Kits (JDKs), Globus Toolkit versions for a variety of essential Globus Toolkit services. The results are displayed in convenient reports in HTML format that may be published on demand to a Web-server. Hence the framework can be used by Grid administrators to perform simple production tests helping to provide a report about the functionality of a Grid.

However this framework can also be used by individual users to test their ability to access Globus Toolkit Services based on a configuration file the user may maintain.

This chapter assumes some familiarity with the Unix command line and Bash scripts.

9.2 Requirements

In order to be able to run the Java CoG Kit tests, you need to be sure that the following are available:[1]

- A Unix-like operating system (BSD, Linux, Solaris, HP-UX)
- Bash
- Concurrent Versions System (CVS)
- Jakarta Ant
- At least one Java Development Kit (1.3.1 or higher)
- GNU wget

[1] We have not spend effort in making this a 100% Java framework.

9.3 Installation

To run the Java CoG Kit tests, all you need is the `nightly-tests` script available from the Java CoG Kit web-site:

Test script : `http://www.globus.org/cog/java/nightly-test`

9.4 Configuration

Configuration of the tests is done by changing the values of the variables found in the beginning of the test script. A detailed list of these variables together with their meaning and sample values is provided below:

`LOCAL` : **Syntax:** `LOCAL = "yes" | "no"`

Specifies whether the sources are going to be fetched from the source repository (”no”) or already downloaded sources will be used (”yes”)

Example: `LOCAL="no"`

`BUILDDIR` : **Syntax:** `BUILDDIR="<directory>"`

If the value of `LOCAL` is `"yes"` this variable represents the location of the Java CoG Kit sources.[2]

If the value of `LOCAL` is `"no"` it points to the directory where the sources will be downloaded by the script. This directory will be created if it does not already exist.

Example: `BUILDDIR="$HOME/tmp/cog-test"`

`HTMLOUTDIR` : **Syntax:** `HTMLOUTDIR="<directory>"`

This variable points to the directory where the output of the tests will go.

Example: `HTMLOUTDIR="$HOME/public_html/tests"`

`JDKSDIR` : **Syntax:** `JDKSDIR="<directory>"`

This variable represents a directory where at least a Java Development Kit can be found. This directory will be searched for valid Java Development Kits. If, inside the `JDKSDIR`, you have a symbolic link pointing to a Java Development Kit directory also within the `JDKSDIR`, that specific Java Development Kit will still only be used once. You can also have directories that contain other things than a Java Development Kit. Such directories will be ignored.

Example: `JDKSDIR="/usr/local"`

`JDKS` : **Syntax:** `JDKS="[<directory> [<directory> [...]]]"`

This variable can be used as an alternative to the `JDKSDIR` variable. It must contain a list of Java Development Kit distribution directories. If you wish to use the `JDKSDIR` method instead, this variable must remain blank.

Example: `JDKS="/usr/local/jdk1.3.1_07 /usr/local/j2sdk1.4.1_01"`

[2] Currently this does not work. If you choose a local test, the script will assume that it was executed from the `ogce/bin` directory.

ANT_HOME : **Syntax:** ANT_HOME="<directory>"

Specifies the location of Jakarta Ant.

Example: ANT_HOME="/usr/local/jakarta-ant-1.5.1"

CVSROOT : **Syntax:** See the CVS manual for details about CVSROOT syntax.

Indicates the location of the source repository of the Java CoG Kit. This variable is only used if you set LOCAL to "no" above. We recommend you leave this variable unmodified. The tests were designed to run without user intervention, and modification of the CVSROOT variable may lead to CVS hanging while waiting for a password.

Caution: Due to the fact that CVS does not allow the password to be specified from the command line even in unsecured mode or when the password is blank, the script uses a hack that will overwrite any passwords that are already locally stored on your machine. The side-effect is that the next time you access a CVS archive for which you had the password stored (in pserver mode), you will have to retype it.

Example: CVSROOT=":pserver:anonymous@cvs.globus.org:/home/dsl/cog/CVS"

COG_PROPERTIES : **Syntax:** COG_PROPERTIES="<file>"

Allows you to specify a cog.properties file to be used for the tests. You can safely leave this blank, in which case the default $HOME/.globus/cog.properties will be used.

Example: COG_PROPERTIES="$HOME/.globus/cog.properties"

HOSTLIST : **Syntax:** HOSTLIST="[<URL> [<URL> [...]]]"

Contains a space-separated list of tables that describe the machines/services/versions to be used for the tests. A detailed description of the table format is provided in Section 9.5.

Example: HOSTLIST="http://www.lpt.usb.com/machines.txt \
file:///tmp/machines2.txt"

TIMEOUT : **Syntax:** TIMEOUT=<integer>

Specifies, in seconds, the time after which a test is killed if it has not terminated. This seems to be necessary since in some instances, while running on IBMJava 2-14, the Java CoG Kit appears to hang indefinitely.

Example: TIMEOUT=300

9.5 Host Table Format

The host table allows you to specify Globus services in a simple manner. A host table is a simple text file. Each row in the table describes a Globus service version. The individual fields are separated using semicolons. The order and meaning of the fields are as follows:

Host name : The name or IP address of the target machine.

Operating System : The operating system running on the target machine. This field is copied into the reports generated by the tests. It has no functional role.

CPU : The type of CPU(s) that the target machine has. This is just an informative field as well.

Available RAM : The amount of available memory. This field has no functional significance.

Service : A Globus service available on the target machine. The format of this field is: `<name> <version>`. The tests will only recognize the following service names: *gram*, *gsiftp*, and *mds*.

Port Number : Indicates the TCP port number on which the service is running.

The following example shows how such a table may look like:

```
hot.mcs.anl.gov;Slackware Linux 8.1 (2.4.18);PIII 866MHz (x2);512 MB;gram 2.2.2;5222
hot.mcs.anl.gov;Slackware Linux 8.1 (2.4.18);PIII 866MHz (x2);512 MB;gram 2.2.4;5224
hot.mcs.anl.gov;Slackware Linux 8.1 (2.4.18);PIII 866MHz (x2);512 MB;gsiftp 2.2.2;6222
hot.mcs.anl.gov;Slackware Linux 8.1 (2.4.18);PIII 866MHz (x2);512 MB;gsiftp 2.2.4;5224
hot.mcs.anl.gov;Slackware Linux 8.1 (2.4.18);PIII 866MHz (x2);512 MB;mds 2.2.2;7222
hot.mcs.anl.gov;Slackware Linux 8.1 (2.4.18);PIII 866MHz (x2);512 MB;mds 2.2.4;7224
cold.mcs.anl.gov;Solaris 9;Sparc 900 MHz (x2);4 GB;gsiftp 2.2.4;6224
```

9.6 Running the Tests

In order to start the tests, all you need to do is run the test script:

```
> chmod +x nightly-test
> ./nightly-test
```

or

```
> bash nightly-test
```

A log file will be created in the location you chose during the configuration. This log file will contain detailed information about the testing process. You may need to check the log in case something goes terribly wrong.

The output produced by the tests will be available in the location specified through the `HTMLOUTDIR` variable. The output directory will contain an `index.html` file which can be opened using a web-browser. The index file will contain links to all the tests performed, together with information about the machine the tests were run on, and the date and time these tests were performed.

A sample image of how one of the reports could look like is provided in Figure 9.1

Clicking on any of the links will either provide help with that item or display additional details.

Java CoG Kit general tests

Host machine: dvorak.mcs.anl.gov
OS: Linux 2.4.18-w1 i686 unknown
Date: Thu Mar 27 12:12:17 CST 2003

IBMJava2-131

Time	Host	OS	GRIS	FTP	GRAM
0:14:43	grieg.mcs.anl.gov	Mandrake 7.2 (2.4.17)	Failed	Ok	Ok
0:14:55	shostakovich.mcs.anl.gov	RedHat 7.3 (2.4.18-w1)	Ok	Ok	Ok
0:15:2	wagner.mcs.anl.gov	Solaris 9	Ok	Ok	Ok
0:15:2	mozart.mcs.anl.gov	Solaris 9	Ok	Ok	Ok
0:15:2	mahler.mcs.anl.gov	Solaris 7	Ok	Ok	Ok
0:15:2	schumann.mcs.anl.gov	IRIX 6.5	Ok	Ok	Ok

IBMJava2-14

Time	Host	OS	GRIS	FTP	GRAM
0:32:7	grieg.mcs.anl.gov	Mandrake 7.2 (2.4.17)	Failed	Ok	Ok
0:32:18	shostakovich.mcs.anl.gov	RedHat 7.3 (2.4.18-w1)	Ok	Ok	Ok
0:32:23	wagner.mcs.anl.gov	Solaris 9	Ok	Ok	Ok
0:32:23	mozart.mcs.anl.gov	Solaris 9	Ok	Ok	Ok
0:32:23	mahler.mcs.anl.gov	Solaris 7	Ok	Ok	Ok
0:32:23	schumann.mcs.anl.gov	IRIX 6.5	Ok	Ok	Ok

j2sdk1.4.0_03

Time	Host	OS	GRIS	FTP	GRAM
0:49:30	grieg.mcs.anl.gov	Mandrake 7.2 (2.4.17)	Failed	Ok	Ok
0:49:38	shostakovich.mcs.anl.gov	RedHat 7.3 (2.4.18-w1)	Ok	Ok	Ok

Figure 9.1: Sample general test report

10 GridAnt: A Client-side Grid Workflow System

This chapter focuses on a sophisticated client-side workflow management system that can orchestrate complex task dependencies. It gives an overview of process workflows and workflow engines. It further describes the applicability of a client-side workflow system for Grid technologies and introduces the functionality of the *GridAnt* workflow system. It provides detailed instructions for the user to install the GridAnt system and other dependent packages. An introductory set of examples is discussed that helps the end-user to understand the working of the GridAnt system. The current version of GridAnt is not an integral part of the Java CoG Kit and requires a separate installation. However, efforts are being made to integrate the GridAnt module in the Java CoG Kit for future releases.

10.1 Introduction

Significant research has been conducted in recent years to automate complex business tasks using sophisticated workflow management tools. Such tools are extremely useful in expressing complicated business activities as a set of independent work units and orchestrating a series of dependencies across these units. In other words, a workflow management system helps in combining a set of specialized tasks by expressing intricate dependencies between these tasks and exposing them as a single complex activity. To the heart of any workflow system is the workflow engine. The workflow engine is a central controller that handles task dependencies, failure recoveries, performance analysis and process synchronization. Most of the work done in workflow management systems concentrate on the business aspects of the workflow. Little consideration is given to the needs of the client in terms of mapping the process flow of the client. In the Grid community it is essential that the Grid-users have such a tool available to their disposal that enable them to orchestrate complex workflows on the fly without substantial help from the service providers. At the same it is also important that such a workflow system does not burden the Grid-user with the intricacies of the workflow system.

With the perspective of the Grid-user in mind, a simple yet powerful client-side workflow management system has been developed and is named as *GridAnt*. GridAnt which makes use of commodity technologies such as *Apache Ant* [1] and XML. The availability of the GridAnt framework provides a much needed functionality for developing and testing Grid applications with the Globus Toolkit 3 (GT3) [18]. GridAnt uses Apache Ant as its workflow engine. Apache Ant is a popular build tool that is extensively used in the Java community. Its current functionality allows the management of complex dependencies and task flows within the project build process. We extend the functionality of Apache Ant by providing customized Ant tasks to access the Grid.

GridAnt proves to be an excellent tool, not only to map complex client-side workflows, but also as a simplistic client to test the functionality of different Grid services. GridAnt will help applications to make a smooth transition from GT2 to GT3. GridAnt is not claimed as a substitution for more sophisticated and powerful

workflow engines that map complex business processes [19, 20, 21, 22]. Nevertheless, applications with simple process flows tightly integrated to work with the Grid technology can benefit from GridAnt without having to endure any complex workflow architectures. The philosophy adopted by the GridAnt project is to use the workflow engine available with Apache Ant and develop a Grid workflow vocabulary on top of it.

10.2 GridAnt Tasks

The following is a partial list of GridAnt tasks that we plan to implement.

gridSetup : The Grid environment setup.

gridAuthenticate : Initializes the proxy certificate to be used by clients.

gridExecute : Executes an arbitrary job on a remote machine using the Java Job Manager service provided by GT3 alpha.

gridCopy : Provides third party file transfers between GridFtp enabled Grid resources using the Reliable File Transfer service provided by GT3.

gridQuery : Provides capabilities to query the service data of different Grid services.

This is a tentative list and is by no means final. Neither have we implemented all of the above tasks. The initial prototype for GridAnt has the functionality for job submission and file transfer. Other tasks are under development. We release the current version as a technology preview in order to obtain feedback and to engage the community in its further development.

10.2.1 gridExecute

The gridExecute task executes an arbitrary job on a Grid resource. It requires the following input parameters (* specifies a mandatory argument).

factorylocation* : Specifies the location of the Java Job Manager factory service available in GT3.[1]

security : Specifies the XML security parameters. Valid options are *xmlSig* and *xmlEnc* for XML signature and XML encryption respectively. The default is XML signature.

delegation : Specifies the parameters for credential delegation for GSI security. Valid options are *full* and *limited* for full delegation and limited delegation respectively. The default id limited delegation.

executable* : Specifies the command to be executed on the Grid resource.

localExecutable : A boolean flag that specifies if the executable resides on the client machine. The default is *false*.

arguments : Specifies the arguments to be provided with the executed command

directory : Specifies the remote directory in which the command is to be executed

[1] This will eventually be an optional parameter and extended by additional optional parameters to more easily specify a task that is GT2 and GT3 portable. The additional parameters are: *server=<hostname> port=<portnumber> provider='GT3'*. We call this formulation the *uniform hosting environment formulation*

environment : Specifies the environment variables to be set prior to the execution of the command.

outputFile : Specifies the file name to which the output must be redirected. If left blank or not specified, the output is streamed to the standard output. By default output is streamed to the standard output.

errorFile : Specifies the file name to which the error messages must be redirected. If left blank, the errors are streamed to the standard error. By default the errors are streamed to the standard error.

redirect : A boolean flag that specifies if the output and error streams are to be redirected to the client. Default value is *true*.

For example, assume we like to schedule a job on the machine *hot.anl.gov* through port 8080:

```
<gridExecute
    factorylocation="http://hot.anl.gov:8080/.../SecureJobManagerFactory"
    security="xmlEnc"
    delegation="full"
    executable="/bin/ls"
    localExecutable="true"
    arguments="-l"
    directory="/home/amin"
    outputFile="myOutput.txt"
    errorFile="myError.txt"
    redirect="false"
/>
```

or in uniform hosting environment notation

```
<gridExecute
    server="hot.anl.gov"
    port="8080"
    provider="GT3"
    executable="/bin/ls"
    arguments="-l"
    directory="/home/amin"
    outputFile="myOutput.txt"
    errorFile="myError.txt"
    redirect="false"
/>
```

10.2.2 gridCopy

The *gridCopy* task performs third party file transfers between grid resources capable of supporting the GridFtp protocol. This task requires the following input arguments (* specifies a mandatory arguments).

factorylocation* : Specifies the location of the Reliable File Transfer factory service.

security : Specifies the XML security parameters. Valid options are *xmlSig* and *xmlEnc* for XML signature and XML encryption respectively. The default is XML signature.

delegation : Specifies the parameters for credential delegation for GSI security. Valid options are *full* and *limited* for full delegation and limited delegation respectively. The default is limited delegation.

fromURL* : Specifies the url of the file to be copied. The url must be in the form gstftp://machineName:portName/absolutePathName.

toURL* : Specifies the url of the destination address. The url must be in the form gsiftp://machineName:portName/absolutePathName.

parallelStreams : Indicates the number of parallel tcp streams desired for the file transfer. Default is 1.

tcpBuffer : Indicates the tcp buffer size desired for the file transfer. Default is 16384.

For example, assume we like to schedule a transfer from machine machine *hot.anl.gov* to machine *cold.anl.gov* through machine rft.anl.gov on port 8080:

```
<gridCopy
   factorylocation="http://rft.anl.gov:8080/.../ReliableTransferFactoryService"
   security="xmlSig"
   delegation="full"
   fromURL="gsiftp://hot.anl.gov/home/amin/from.txt"
   toURL="gsiftp://cold.anl.gov/home/amin/to.txt"
   parallelStreams="3"
/>
```

or in uniform hosting environment formulation notation

```
<gridCopy
   server="hot.anl.gov"
   port="8080"
   provider="GT3"
   fromURL="gsiftp://[gridftpServer]/home/amin/from.txt"
   toURL="gsiftp://[gridftpServer]/home/amin/to.txt"
   parallelStreams="3"
/>
```

10.3 Installation

We are following the GT3 development to provide a set of tasks that can be orchestrated with GT3 Grid services. The following are the tools required in order to use the GridAnt framework for GT3.

- Java 1.3.1. The GridAnt system also works with Java 1.4, however it requires certain additional configuration for the new security libraries. If you intend to use Java 1.4.0, you will have to copy the Xalan.jar available in the gridant/lib directory to j2sdk1.4.0/jre/lib/endorsed/ directory.
- Apache Ant 1.5 [1].
- Java Cog Kit [**?**].
- GT3 alpha2 Server side components [18]. Specifically, you will need the Java Job Manager service in the *program_execution* module and the Reliable File Transfer service in the *data_management* module.

To install GridAnt you need to checkout the latest source code (compatible with GT3 alpha2) in the cvs repository.

```
> mkdir cog
> cd cog
> cvs -d :pserver:anonymous@cvs.globus.org:/home/dsl/cog/CVS co gt3
> cd gt3/gridant
> ant build
```

Note: To install the GridAnt components for GT3 alpha, use:

```
> cvs -d :pserver:anonymous@cvs.globus.org:/home/dsl/cog/CVS co -r alpha gt3
```

10.4 Security

GridAnt uses the Grid Security Infrastructure (GSI) for authentication, authorization, and credential delegation. Please refer to Chapter 4 for a detailed description on obtaining the required credentials and the initial setup to make GridAnt GSI compliant.

10.5 Examples

Several examples are available in the build.xml file.

10.5.1 gridExecute

To test the *gridExecute* GridAnt task:

```
> cd gt3/gridant
> ant build
> ... create your proxy certificate
> ... start the GT3 service container
> ... edit the build.xml in the gridant directory such that the
          arguments in that target "submitDemo" reflect the
          appropriate values ...
> ant submitDemo
```

To test a simple GUI client for Job submission:

```
> cd gt3/gridant
> ant build
> ... create your proxy certificate ...
> ... start the GT3 service container ...
> ant submitGUI
> ... make the necessary entries in the GUI and submit the job ...
```

10.5.2 gridCopy

To test the *gridCopy* GridAnt task:

```
> cd gt3/gridant
> ant build
> ... create your proxy certificate ...
> ... start the GT3 service container ...
> ... edit the build.xml in the gridant directory such that the
      arguments in that target "rftDemo" reflect the appropriate
      values ...
> ant rftDemo
```

To test a simple GUI client for FileTransfer:

```
> cd gt3/gridant
> ant build
> ... create your proxy certificate ...
> ... start the GT3 service container ...
> ant rftGUI
> ... make the necessary entries in the GUI and submit the
      file transfer ...
```

10.6 Complex Example

To be completed ...

Appendix A Program Options

A.1 globus2jks

The command `globus2jks` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java KeyStoreConvert options
        java KeyStoreConvert -help

        Converts Globus credentials (user key and certificate) into
        Java keystore format (JKS format supported by Sun).

        Options
        -help | -usage
                Displays usage.
        -version
                Displays version.
        -debug
                Enables extra debug output.
        -cert     <certfile>
                Non-standard location of user certificate.
        -key      <keyfile>
                Non-standard location of user key.
        -alias    <alias>
                Keystore alias entry. Defaults to 'globus'
        -password <password>
                Keystore password. Defaults to 'globus'
        -out      <keystorefile>
                Location of the Java keystore file. Defaults to
                'globus.jks'
```

A.2 globus-gass-server

The command `globus-gass-server` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java GassServer options
        java GassServer -version
        java GassServer -help

        Options
        -help | -usage
                Displays usage
        -s | -silent
                Enable silent mode (Don't output server URL)
        -r | -read
                Enable read access to the local file system
         -w | -write
                Enable write access to the local file system
```

```
 -o
        Enable stdout redirection
-e
        Enable stderr redirection
-c | -client-shutdown
        Allow client to trigger shutdown the GASS server
        See globus-gass-server-shutdown
-p <port> | -port <port>
        Start the GASS server using the specified port
-i | -insecure
        Start the GASS server without security
-n <options>
        Disable <options>, which is a string consisting
        of one or many of the letters "crwoe"
```

A.3 globus-gass-server-shutdown

The command `globus-gass-server-shutdown` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java GassServerShutdown -usage -version <GASS-URL>
        java GassServerShutdown -help

        Allows the user to shut down a (remotely) running
        GASS server, started with client-shutdown permissions
        (option -c).

        Options:
        -help | -usage
                Displays usage
        -version
                Displays version
```

A.4 globus-personal-gatekeeper

The command `globus-personal-gatekeeper` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java Gatekeeper options
        java Gatekeeper -version
        java Gatekeeper -help

        Options
        -help | -usage
                Displays usage
        -p | -port
                Port of the Gatekeeper
        -d | -debug
                Enable debug mode
        -s | -services
                Specifies services configuration file.
         -l | -log
                Specifies log file.
        -gridmap
```

```
        Specifies gridmap file.
-proxy
        Proxy credentials to use.
-serverKey
        Specifies private key (to be used with -serverCert.
-serverCert
        Specifies certificate (to be used with -serverKey.
-caCertDir
        Specifies locations (directory or files) of trusted
        CA certificates.
```

A.5 globusrun

The command `globusrun` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java GlobusRun options RSL String
        java GlobusRun -version
        java GlobusRun -help

        Options
        -help | -usage
                Display help.
        -v | -version
                Display version.
        -f <rsl filename> | -file <rsl filename>
                Read RSL from the local file <rsl filename>. The RSL
                must be a single job request.
        -q | -quiet
                Quiet mode (do not print diagnostic messages)
        -o | -output-enable
                Use the GASS Server library to redirect standout output
                and standard error to globusrun. Implies -quiet.
        -s | -server
                $(GLOBUSRUN_GASS_URL) can be used to access files local
                to the submission machine via GASS. Implies
                -output-enable and -quiet.
        -w | -write-allow
                Enable the GASS Server library and allow writing to
                GASS URLs. Implies -server and -quiet.
        -r <resource manager> | -resource-manager <resource manager>
                Submit the RSL job request to the specified resource
                manager. A resource manager can be specified in the
                following ways:
                 - host
                 - host:port
                 - host:port/service
                 - host/service
                 - host:/service
                 - host::subject
                 - host:port:subject
                 - host/service:subject
                 - host:/service:subject
                 - host:port/service:subject
                For those resource manager contacts which omit the port,
                service or subject field the following defaults are used:
                port = 2119
                service = jobmanager
                subject = subject based on hostname
                This is a required argument when submitting a single RSL
                request.
```

```
-k | -kill <job ID>
        Kill a disconnected globusrun job.
-status <job ID>
        Print the current status of the specified job.
-b | -batch
        Cause globusrun to terminate after the job is successfully
        submitted, without waiting for its completion. Useful
        for batch jobs. This option cannot be used together with
        either -server or -interactive, and is also incompatible
        with multi-request jobs. The "handle" or job ID of the
        submitted job will be written on stdout.
-stop-manager <job ID>
        Cause globusrun to stop the job manager, without killing
        the job. If the save_state RSL attribute is present, then a
        job manager can be restarted by using the restart RSL
        attribute.
-fulldelegation
        Perform full delegation when submitting jobs.

Diagnostic Options
-p | -parse
        Parse and validate the RSL only. Does not submit the job
        to a GRAM gatekeeper. Multi-requests are not supported.
-a | -authenticate-only
        Submit a gatekeeper "ping" request only. Do not parse the
        RSL or submit the job request. Requires the
        -resource-manger argument.
-d | -dryrun
        Submit the RSL to the job manager as a "dryrun" test
        The request will be parsed and authenticated. The job
        manager will execute all of the preliminary operations,
        and stop just before the job request would be executed.

Not Supported Options
-n | -no-interrupt
```

A.6 globus-url-copy

The command `globus-url-copy` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java GlobusUrlCopy options fromURL toURL
        java GlobusUrlCopy -help

        Options
        -s  <subject> | -subject <subject>
              Use this subject to match with both the source
              and destination servers
        -ss <subject> | -source-subject <subject>
              Use this subject to match with the source server
        -ds <subject> | -dest-subject <subject>
              Use this subject to match with the destination server
        -notpt | -no-third-party-transfers
              Turn third-party transfers off (on by default)
        -nodcau | -no-data-channel-authentication
              Turn off data channel authentication for ftp transfers
              Applies to FTP protocols only.

        Protocols supported:
        - gass (http and https)
        - ftp (ftp and gsiftp)
```

```
- file
```

A.7 grid-cert-info

The command `grid-cert-info` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java CertInfo -help -file certfile -all -subject ...

        Displays certificate information. Unless the optional
        file argument is given, the default location of the file
        containing the certficate is assumed:

          -- /home/mike/.globus/usercert.pem

        Options
        -help | -usage
                Display usage.
        -version
                Display version.
        -file certfile
                Use 'certfile' at non-default location.
        -globus
                Prints information in globus format.

        Options determining what to print from certificate

        -all
                Whole certificate.
        -subject
                Subject string of the cert.
        -issuer
                Issuer.
        -startdate
                Validity of cert: start date.
        -enddate
                Validity of cert: end date.
```

A.8 grid-change-pass-phrase

The command `grid-change-pass-phrase` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java ChangePassPhrase -help -version -file private_key_file

        Changes the passphrase that protects the private key. If the
        -file argument is not given, the default location of the file
        containing the private key is assumed:

          -- /home/mike/.globus/userkey.pem

        Options
        -help | -usage
                Display usage.
        -version
```

```
        Display version.
    -file location
        Change passphrase on key stored in the file at
        the non-standard location 'location'.
```

A.9 grid-info-search

The command `grid-info-search` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
grid-info-search  options  <search filter> attributes

    Searches the MDS server based on the search filter, where some
    options are:
       -help
            Displays this message

       -version
            Displays the current version number

       -mdshost host (-h)
            The host name on which the MDS server is running
            The default is none.

       -mdsport port (-p)
            The port number on which the MDS server is running
            The default is 2135

       -mdsbasedn branch-point (-b)
            Location in DIT from which to start the search
            The default is 'mds-vo-name=local, o=grid'

       -mdstimeout seconds (-T)
            The amount of time (in seconds) one should allow to
            wait on an MDS request. The default is 30

       -anonymous (-x)
            Use anonymous binding instead of GSSAPI.

     grid-info-search also supports some of the flags that are
     defined in the LDAP v3 standard.
     Supported flags:

      -s scope   one of base, one, or sub (search scope)
      -P version protocol version (default: 3)
      -l limit   time limit (in seconds) for search
      -z limit   size limit (in entries) for search
      -Y mech    SASL mechanism
      -D binddn  bind DN
      -v         run in verbose mode (diagnostics to standard output)
      -O props   SASL security properties (auth, auth-conf, auth-int)
      -w passwd  bind password (for simple authentication)
```

A.10 grid-proxy-destroy

The command `grid-proxy-destroy` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java ProxyDestroy -dryrun file1...
        java ProxyDestroy -help

        Options
        -help | -usage
                Displays usage
        -dryrun
                Prints what files would have been destroyed
        file1 file2 ...
                Destroys files listed
```

A.11 grid-proxy-info

The command `grid-proxy-info` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java ProxyInfo options
        java ProxyInfo -help

        Options:
        -help | usage
                Displays usage.
        -file <proxyfile>  (-f)
                Non-standard location of proxy.
        printoptions
                Prints information about proxy.
        -exists options  (-e)
                Returns 0 if valid proxy exists, 1 otherwise.
        -globus
                Prints information in globus format

        printoptions
        -subject
                Distinguished name (DN) of subject.
        -issuer
                DN of issuer (certificate signer).
        -identity
                DN of the identity represented by the proxy.
        -type
                Type of proxy.
        -timeleft
                Time (in seconds) until proxy expires.
        -strength
                Key size (in bits)
         -all
                All above options in a human readable format.
        -text
                All of the certificate.
        -path
                Pathname of proxy file.

        options to -exists (if none are given, H = B = 0 are assumed)
```

```
-hours H     (-h)
        time requirement for proxy to be valid.
-bits  B     (-b)
        strength requirement for proxy to be valid
```

A.12 grid-proxy-init

The command `grid-proxy-init` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java ProxyInit options
        java ProxyInit -help

        Options:
        -help | -usage
                Displays usage.
        -version
                Displays version.
        -debug
                Enables extra debug output.
        -verify
                Verifies certificate to make proxy for.
        -quiet | -q
                Quiet mode, minimal output
        -limited
                Creates a limited proxy.
        -independent
                Creates a independent globus proxy.
        -old
                Creates a legacy globus proxy.
        -hours H
                Proxy is valid for H hours (default:12).
        -bits  B
                Number of bits in key {512|1024|2048|4096}.
        -globus
                Prints user identity in globus format.
        -policy <policyfile>
                File containing policy to store in the
                ProxyCertInfo extension
        -pl <oid>
                OID string for the policy language.
        -policy-language <oid>
                used in the policy file.
        -path-length <l>
                Allow a chain of at most l proxies to be
                generated from this one
        -cert     <certfile>
                Non-standard location of user certificate
        -key      <keyfile>
                Non-standard location of user key
        -out      <proxyfile>
                Non-standard location of new proxy cert.
        -pkcs11
                Enables the PKCS11 support module. The
                -cert and -key arguments are used as labels
                to find the credentials on the device.
```

A.13 myproxy

The command `myproxy` can be found in the `build/cog-1.1a/bin` directory. This commandline provides a convenient wrapper to a Java class. Thus, it has the same options as listed below.

```
Syntax: java MyProxy options command
        java MyProxy -version
        java MyProxy -help

        Options
        -help
                Displays usage
        -v | -version
                Displays version

        -h <host> | -host <host>
                Hostname of the myproxy-server
        -p <port> | -port <port>
                Port of the myproxy-server
                (default 7512)
        -l <username> | -username <username>
                Username for the delegated proxy
        -t <time> | -portal_lifetime <time>
                Lifetime of delegated proxy on
                the portal (default 2 hours)
        -c <time> | -cred_lifetime <time>
                Lifetime of delegated proxy
                (default 1 week - 168 hours)
                Note: Only used by PUT operation
        -s <subject> | -subject <subject>
                Performs subject authorization
        command
                One of the following:
                        put     - put proxy
                        get     - get proxy
                        anonget - get proxy without local credentials
                        destroy - remove proxy
                        info    - credential information
```

Appendix B Command overview

B.1 New Format for the table

The main ant build file for building the Java CoG Kit is `build.xml`. The help target present in each of the xml files gives the details of all the targets supported and their functionality.

The `demos.xml` contain all the gui demos present in ogce and tools.xml contains targets for running the command line tools using ant. The following table gives an overview of the equivalent ant targets which are available for each of the scripts present in the `<cog-install-path>/bin` directory in the alphabetical order.

Command	Buildfile	target	Section
globus-gass-server-shutdown	tools.xml	globus-gass-server-shutdown	A.3
globus-gass-server	tools.xml	globus-gass-server	A.2
globus-personal-gatekeeper	tools.xml	globus-personal-gatekeeper	A.4
globus-url-copy	tools.xml	globus-url-copy	A.6
globus2jks	N/A	N/A	A.1
globusrun	tools.xml	globusrun	A.5
grid-cert-info	tools.xml	grid-cert-info	A.7
grid-change-pass-phrase	N/A	N/A	A.8
grid-info-search	tools.xml	grid-info-search	A.9
grid-proxy-destroy	tools.xml	grid-proxy-destroy	A.10
grid-proxy-info	tools.xml	grid-proxy-info	A.11
grid-proxy-init	N/A	N/A	A.12
hellogridftp	demos.xml	N/A	N/A
helloworld	demos.xml	N/A	N/A
myproxy	tools.xml	myproxy	A.13
ogce-setup	demos.xml	setup	N/A
setup	demos.xml	old-setup	N/A
visual-grid-proxy-init	demos.xml	login	N/A

Appendix C Frequently Asked Questions

C.1 Installation

- What are the requirements for the Java CoG Kit?
  Section [3.2]
- How do I download the stable distribution?
  Section [3.4]
- How do I download the development distribution?
  Section [3.4.5]
- How do I compile the Java CoG Kit sources?
  Section [3.5]
- How do I configure the Java CoG Kit?
  Section [3.6]
- A script complains about COG_INSTALL_PATH not being set; why?
  Section [3.6.1]
- A program complains about a missing proxy/certificate; why?
  Section [3.6.3]

C.2 Security

C.2.1 General Grid security Questions

- Why is Grid security so important?
  Section [4.1.1]
- What is the difference between a normal UNIX/Windows username/pasword to the Grid security infrastrucure?
  Section [4.1.1]
- What is a certificate?
  Section [4.1.2]
- What is a CA?
  Section [4.1.2]
- What is a proxy?
  Section [4.1.3]
- What is the difference between a certificate and a proxy?
  Section [4.1.3]
- What needs to be protected from others and how?
  Section [4.2.6]
- What is a gridmap file?
  Section [4.2.5]

- What is MyProxy?
  Section [4.3]
- Can the The Java CoG Kit work behind a firewall? How do I limit the range of ports that the Java CoG Kit will use?
  Section [4.5]

C.2.2 Questions related to user certificates and certificate authority

- How do I aquire a certificate?
  Section [4.2.1]
- How do I renew a certificate?
  Section [4.2.3]
- How do I change the pass-phrase?
  Please see the grid-change-pass-phrase tool in Section [4.4.2] and [4.4.3]
- How do I get the CA's certificate?
  Section [4.2.4]
- What information can I get from a certificate? How?
  Please see the grid-cert-info tool in Section [4.4.2] and [4.4.3]

C.2.3 Questions related to proxy certificates

- How do I create/renew/destroy a proxy?
  Please see the tools described in Section [4.4.2] and [4.4.3]
- How do I get information about my proxy?
  Please see the grid-proxy-info tool in Section [4.4.2] and [4.4.3]

C.2.4 Questions related to host certificates and gridmap files

- How do I get added to the gridmap file?
  Section [4.2.5]

C.2.5 MyProxy

- how to store/retrieve credentials to myProxy?
  Please see the myproxy tool in Section [4.4.2] and [4.4.3]

C.2.6 Miscellaneous

- How do I configure the Java CoG Kit with the security files?
  Please see the "Java CoG Kit configuration wizard" in Section [4.4.1]
- Does the Java CoG Kit provide API for security-related tasks? How do I use them?
  Section [4.4.5]
- Portability and design issues regarding the security API in previous versions of the Java CoG Kit and in version 1.1a
  Section [4.4.5]

- I'm getting following error when connecting to a gatekeeper: "Server certificate rejected by ChainVerifier." What does it mean and how can I fix it?

  In most cases, it means that the client either does not trust or does not have the CA certificate that signed the server certificate. Please see the Section [4.2.4].

- I'm getting following error when connecting to a server: "Handshake failure." What does it mean and how can I fix it?

  Probably you have a proxy not compatible with the server. Please see the `grid-proxy-init` tool described in Section 4.4.2

C.3 File I/O and Transfer

C.3.1 Overview

- What are the issues involved in file transfer over the Grid?
  Section [5.1.1]
- What is GridFTP?
  Section [5.1.1] and [5.1.2]
- What is GASS?
  Section [5.1.1] and [5.1.3]
- Can I still use FTP and SCP?
  Section [5.1.4]
- What is the difference between GridFTP and GASS?
  Section [5.1.1]
- What is a third-party transfer?
  See the GridFTP section : [5.1.2]
- What are parallel and striped transfers?
  See the GridFTP section : [5.1.2]
- Are GridFTP and GASS standard services that run on every Globus-enabled resource?
  See the GridFTP Section [5.1.2]) and GASS section [5.1.3]
- Is there a provision to monitor the progress of a transfer and restart it if fails?
  Yes. See "restart markers" mentioned in the GridFTP section : [5.1.2]

C.3.2 GridFTP

- How do I store and retrieve files using GridFTP?
  Different methods of doing this are described in Section [5.2]
- How do I transfer files between two GridFTP servers?
  Different methods of doing this are described in Section [5.2]
- How do I monitor progress of my transfers?
  Currently you can do this only using the GridFTP APIs. Please check the GridFTP Programmer's Guide available at http://www.globus.org/cog/jftp/guide.html

- How do I come to know if my transfer fails? How do I restart it?
  Currently you can do this only using the GridFTP APIs. Please check the GridFTP Programmer's Guide available at http://www.globus.org/cog/jftp/guide.html
- Why am I not able to obtain a list of files in a directory on a FTP/GridFTP server?
  Section [5.2.8]
- I'm getting the following error when I am trying to transfer a file or do a file listing: "425 Can't build data connection: Connection refused." What does it mean and how can I fix it?

  Your computer may be behind a firewall that does not allow the data connection. Please see the "GridFTP data channels" paragraph in Section [4.5].

C.3.3 GASS

- How do I start a GASS server on my local machine?
  Different methods of doing this are described in Section [5.3]
- How do I start a GASS server on a remote machine?
  Section [5.3.5]
- How do I store and retrieve files using GASS?
  Different methods of doing this are described in Section [5.3]
- How do I transfer files between two machines using GASS?
  Different methods of doing this are described in Section [5.3]

C.3.4 Version differences

- What are the differences in file transfer APIs between CoG 0.9.13 and later versions?

C.4 Job Execution

C.4.1 GRAM

- What is GRAM?
  Section [6.1]
- What is a Gatekeeper?
  Section [6.1.1]
- What is a job manager?
  Section [6.1.2]
- What is file staging?
  Section [6.1.4]
- What is RSL?
  Section [6.2]
- How do I start a job from command line?
  Section [6.3.2] and Section [6.3.3]

- How do I use the Java CoG Kit API to start a job?
  Section[6.3.5]
- What are interactive/batch jobs? when to chose what?
  Section [6.1.3]
- How do I stream output/errors back to my machine?
  Section [6.3.5]

C.5 Grid Information Service

C.5.1 General

- What is a directory service?
  Section [7.1]
- What is MDS?
  Section [7.1]
- What are the differences between Java CoG Kit and Globus grid information search tool?
  Section [7.8]
- Where do I get detailed information for Globus MDS?
  Section [7.1]

C.5.2 Architecture of MDS

- What are the main components in Mds?
  Section [7.2]
- What is GRIS ?
  Section [7.2]
- What is the functionality of GIIS?
  Section [7.2]
- How do GRIS and GIIS interact with each other?
  Section [7.2]
- What are the different kinds of information you can retrive using MDS?
  Section [7.2]
- What is the architecture of the MDS?
  Section [7.2]
- What is an information provider?
  Section [7.2]
- Can I use GRIS without a GIIS?
  Section [7.2]
- Where do I find MDS information providers?
  Section [7.2]

C.5.3 Security in MDS

- How GSI work with MDS?
  Section [7.3]
- What is SASL authentication in regards to MDS?
  Section [7.4.4]
- Are there any site policies attached with GIIS and GRIS?
  Section [7.3.1]
- Can I share information that I got from GRIS?
  Section [7.3.1]

C.5.4 Retrieving information from MDS

- How do I retrieve MDS information using the command line tool?
  Section [7.4.2] and Section [7.4.3]
- How do I invoke the MDS services using API?
  Section [7.4.4]
- Do I have any problems using Netscape library to access MDS with GSI authentication?
  Section [7.4.4]
- How can I hook up to the GSI security using JNDI?
  Section [7.4.4]
- Is the LDAP browser integrated with the Java CoG?
  Section [7.4.1]
- How do I choose selecting between the JNDI and netscape SDK?
  Section [7.4.4]
- Why should I not keep a connection to the MDS for a long time?
  Section [7.6.1]

C.5.5 Performace Issues with MDS

- What is the difference in updating quality for GIIS and GRIS ?
  Section [7.6]
- Can I write GRIS and GIIS in java?
  Section [7.6]
- What performace can I expect from MDS?
  Section [7.6]
- Does the performance of GRIS affect the performance of GRIS?
  Section [7.6]

C.6 Server Side Java CoG Kit

C.6.1 General

- Does Java CoG Kit provide any server side implementations?
  Section [8.1]

- What are the server side functionalities provided by Java CoG Kit?
  Section [8.1]
- Where do I find detailed information regarding the Globus server side functionalities?
  Section [8.1]

C.6.2 Job Execution Service

- What is a personal gatekeeper?
  Section [8.2]
- What is a Job Manager Service?
  Section [8.2]
- How do I configure my personal gatekeeper?
  Section [8.2.1]
- Are there any limitations in Java CoG Kit Job Execution Service?
  Section [8.2.2]
- Can I allow mulitple uses to access my personal gatekeeper and submit their jobs to it?
  Section [8.2]
- What are the different ways of starting up the personal gatekeeper?
  Section [8.2.3]
- Are there any features the gatekeeper does not support when compared with the Globus Personal Gatekeeper implementation?
  Section [8.2.4]

C.6.3 GASS server

- Can I transfer data using Java CoG Kit ?
  Section [8.3]
- What is GASS?
  Section [8.3]
- What is the protocol does GASS use to transfer data?
  Section [8.3]
- Are there any limitations in Java CoG Kit GASS implementation?
  Section [8.3.1]
- Does Java CoG GASS support cache management?
  Section [8.3.1]
- Are there any features GASS in Java CoG Kit does not support when compared with the Globus implementation?
  Section [8.3.3]
- What are the various ways I can start up the GASS server?
  Section [8.3.2]

C.7 GridAnt

- What is a process workflow ?
  Section [10.1]
- What is the difference between server-side and client-side workflows ?
  Section [10.1]
- What workflow engine is used in GridAnt, and why ?
  Section [10.1]
- What is the list of tasks that GridAnt must implement ?
  Section [10.2]
- What is the current status of GridAnt ?
  Section [10.2]
- What version of Java is required for GridAnt ?
  Section [10.3]
- What version of Ant is required for GridAnt ?
  Section [10.3]
- What version of GT3 is required for GridAnt ?
  Section [10.3]
- How do I setup GT3 modules to work with GridAnt ?
  Section [10.3]
- How do I setup the Java CoG Kit to to work with GridAnt ?
  Section [3.2]
- How do I execute a remote job with GridAnt ?
  Section [10.5]
- How do I execute a third party reliable file transfer with GridAnt ?
  Section [10.5]
- How is GridAnt compatible with GT2 ?
  Yet to describe

Bibliography

[1] "Ant – a Java-based Build Tool," Web Page. [Online]. Available: http://ant.apache.org 17, 82, 85

[2] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," in *5th ACM Conference on Computer and Communications Security*. ACM Press, Nov. 2-5 1998, pp. 83–92. [Online]. Available: ftp://ftp.globus.org/pub/globus/papers/security.pdf 25

[3] J. Novotny, S. Tuecke, and V. Welch, "An Online Credential Repository for the Grid: MyProxy," in *Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10)*. San Francisco: IEEE Press, Aug. 2001. [Online]. Available: http://www.globus.org/research/papers/myproxy.pdf 25, 29

[4] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996. 25

[5] "Grid Security Infrastructure," Web Page. [Online]. Available: http://www.globus.org/security/ 25

[6] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, and S. Tuecke, "GridFTP Protocol Specification," Web Page, September 2002. [Online]. Available: http://www.globus.org/research/papers/GridftpSpec02.doc 38, 39

[7] W. Allcock, I. Foster, and S. Tuecke, "Protocols and Services for Distributed Data-Intensive Science," in *ACAT2000 Proceedings*, Fermi National Accelerator Laboratory. Chicago, Oct. 16-20 2000, pp. 161–163, http://www.globus.org/research/papers/ACAT3.pdf. 38

[8] "GridFTP," Web Page. [Online]. Available: http://www.globus.org/datagrid/gridftp.html 38

[9] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, "GASS: A Data Movement and Access Service for Wide Area Computing Systems," in *Proceedings of IOPADS'99*. Atlanta, Georgia: ACM Press, May 1999. [Online]. Available: ftp://ftp.globus.org/pub/globus/papers/gass.pdf 38

[10] "Globus Access to Secondary Storage," Web Page. [Online]. Available: http://www.globus.org/gass/ 38, 53

[11] "Dsniff: A Tool for Penetration Testing," Web Page. [Online]. Available: http://naughty.monkey.org/~dugsong/dsniff/ 40

[12] B. Allcock and R. Madduri, "Reliable File Transfer Service," Web Page. [Online]. Available: http://www-unix.globus.org/ogsa/docs/alpha3/services/reliable_transfer.html 40, 42

[13] "RFC 2228: FTP Security Extensions," Web Page. [Online]. Available: http://www.ietf.org/rfc/rfc2228.txt 46

[14] “GRAM Job Manager Reference Manual.” [Online]. Available: http://www.globus.org/api/c-globus-2.2/globus_gram_job_manager/html/main.html 52

[15] “The Monitoring and Discovery Service,” Web Page. [Online]. Available: http://www.globus.org/mds 60

[16] G. von Laszewski, S. Fitzgerald, I. Foster, C. Kesselman, W. Smith, and S. Tuecke, “A Directory Service for Configuring High-Performance Distributed Computations,” in *Proceedings of the 6th IEEE Symposium on High-Performance Distributed Computing*, 5-8 Aug. 1997, pp. 365–375. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/fitzgerald--hpdc97.pdf 60

[17] “Globus Toolkit 2.2 MDS Technology Brief,” Web Page. [Online]. Available: http://www.globus.org/mds/mdstechnologybrief_draft4.pdf 60

[18] “Open Grid Services Architecture (OGSA),” Web Page. [Online]. Available: http://www.globus.org/ogsa 82, 85

[19] “DAGMan (Directed Acyclic Graph Manager),” Web Page. [Online]. Available: http://www.cs.wisc.edu/condor/dagman/ 83

[20] “BPEL4WS: Business Process Execution Language for Web Services Version 1.0,” Web Page. [Online]. Available: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel 83

[21] “XLANG: Web Services for Business Process Design,” Web Page. [Online]. Available: http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm 83

[22] “Web Services Flow Language (WSFL),” Web Page. [Online]. Available: www.ibm.com/software/solutions/webservices/pdf/WSFL.pdf 83

[23] G. von Laszewski, I. Foster, J. Gawor, and P. Lane, “A Java Commodity Grid Kit,” *Concurrency and Computation: Practice and Experience*, vol. 13, no. 8-9, pp. 643–662, 2001. [Online]. Available: http://www.mcs.anl.gov/~gregor/papers/vonLaszewski--cog-cpe-final.pdf

Index

Acknowledgments, 16
Administrative Contact, 16
ant, 17

Bugs, 13

Clock synchronisation, 23
cog.properties, 23
Command
    globus-gass-server, 48, 75
    globus-gass-server-shutdown, 48
    globus-personal-gatekeeper, 73
    globus-url-copy, 48
    globusrun, 56
    grid-info-search, 62
Commands
    globus-url-copy, 42
Contact, 16
Contributors, 16

FAQ, 97
    File I/O, 99
    GASS Server, 103
    GRAM Server, 103
    GridAnt, 104
    gridFTP, 99
    Information Service, 101
    Installation, 97
    Job Execution, 100
    MDS, 101
    Security, 97
    Transfer, 99
File I/O, 38
FIle Transfer
    Third-party, 45
File Transfer, 38
File Transfer GUI, 40

GASS, 39
Gatekeeper, 52
GIIS, 61
GIS, 60
    Performance, 69
    schema, 68
    Use Cases, 69
globus-gass-server, 48, 75
globus-gass-server-shutdown, 48
globus-personal-gatekeeper, 73
globus-url-copy, 42, 48
globusrun, 56
grid-info-search, 62
GridAnt, 82
    gridCopy, 84
    gridExecute, 83
    Installation, 85
    Security, 86
    Tasks, 83
GridFTP, 38
    API, 43
GRIS, 60
GUI
    File Transfer, 40
    Job Submission, 55
        Desktop, 56
        Form, 55
    LDAP Browser, 62

Installation, 17
    Clock synchronisation, 23
    cog.properties, 23
    configuration, 23
IPs, 61

JNDI
    Anonymous, 65
    Authenticate, 67
Job Execution Service, 72
Job Manager, 52
Job Submission, 52
    API, 57

LDAP Browser, 62
License, 8
    bouncycastle, 11
    cryptix, 11
    Globus Toolkit, 9
    GPTL, 9
    junit, 11
    log4j, 11
    puretls, 11
    soaprmi11, 11
    xerces, 11
    xml4j, 11

Mailing List, 13
MDS, 60

Nescape SDK
    Anonymous, 65
Netscape SDK
    Authenticate, 67

Production testing, 77
Project Registration, 8

RSL, 53

Schema, 68
Server, 72

Testing, 77
Third-party transfer, 45

Website, 13